Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías Industriales

Desarrollo de Librería para Posicionador Basado en Motor Paso a Paso

Autor: Gonzalo Escobar Granja

Tutor: Luis Fernando Castaño Castaño

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Desarrollo de Librería para Posicionador Basado en Motor Paso a Paso

Autor:
Gonzalo Escobar Granja

Tutor:
Luis Fernando Castaño Castaño
Doctor Ingeniero Industrial

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo de Fin de Grado: Desarrollo de Librería para Posicionador Basado en Motor Paso a Paso

Autor: Gonzalo Escobar Granja

Tutor: Luis Fernando Castaño Castaño

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

Agradecimientos

A D.Fernando Castaño Castaño por su iluminación en el campo de la automatización.
Han sido muchas horas de inconmensurable paciencia y ayuda.

A mis padres, nunca podré agradecerles lo suficiente su apoyo incondicional.

A mi familia.

Os quiero.

Resumen

Los fabricantes de autómatas programables industriales distribuyen el software de control junto con, cada vez más, herramientas a disposición del desarrollador. Pero, ya sea por desconocimiento de la existencia de manuales, por malos hábitos de programación o por falta de ejemplos prácticos, se desaprovecha el gran potencial que prestan estas herramientas.

En este proyecto se trata de arrojar luz sobre estas herramientas, desarrollando un documento de consulta adecuado para la creación, análisis y uso de librerías por parte de programadores de autómatas industriales noveles, proponiendo un enfoque práctico sobre una instalación de formación situada en el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla.

Con las premisas anteriores y la bajo versión 4.0 del software de programación de control, puesta a punto y explotación de autómatas de Schneider Electric: Unity Pro XL, se presenta la siguiente documentación. Ésta se encuentra seccionada en los elementos que conforman las herramientas de depuración y en la escenificación de éstos en un sistema real, como segunda parte de esta memoria.

Para la implementanción del caso práctico se cuenta con el PLC modelo Modicom M340 de Schneider Electric. Se realizará el desarrollo de la automatización de una instalación de alimentación de piezas a una célula de fabricación flexible. La implementación, realizada en los distintos lenguajes de automatismos, está enfocada en la creación de librerías mediante funciones, concebidas con la finalidad principal de su reutilización en otros equipos de la instalación.

Índice

Índice de Tablas

Índice de Figuras

1 ANTECEDENTES Y EVOLUCIÓN HISTÓRICA DE LOS PLC

Con el fin de un aumento de la productividad y la competitividad, las empresas productoras de bienes de consumo y servicio impulsaron el desarrollo y la implementación de nuevas tecnologías. También buscaban relegar procesos de producción y fabricación, históricamente desarrollados en ambientes peligrosos, a elementos insensibles a ellos, adhiriendo a esta idea el concepto de calidad constante en la producción así como la mejora de los costes de producción, se puede afirmar que con esta amalgama de problemas surge el concepto de automatización.

Desde una perspectiva histórica, Ford y General Motors impusieron en 1968 a sus proveedores de automatismos especificaciones para la realización de un sistema electrónico. Este equipo no debía de depender de ordenadores sino ser programable de forma que los usuarios pudieran configurar cada dispositivo que lo conformase para resolver un determinado problema. Estos dispositivos fueron los relés electromecánicos, los cuales en el futuro, posibilitaron la aparición de contadores, registros de desplazamiento, memorias… y como resultado se obtuvo el primer autómata programable.

Debido al gran tamaño requerido por los armarios eléctricos y a la elevada cantidad de stock necesaria para las reparaciones, la productividad se veía penalizada en demasía. La solución se obtuvo gracias a la aparición de los semiconductores de selenio, germanio y silicio, que dieron lugar a los primeros circuitos integrados de puertas lógicas. El avance de la tecnología del silicio hizo posible la posibilidad de disponer de chips que incorporan, en pequeñas áreas, millones de transistores que conforman bloques funcionales que permiten ejecutar gran número de operaciones lógicas y aritméticas.

El primer autómata fabricado para la industria del automóvil fue el modelo MODICON 084 y es considerado el primer PLC comercializado. Estaba basado en semiconductores, en un procesador fundamentado en circuitos integrados y programado con un lenguaje que emulaba los circuitos de relés y la lógica cableada, consiguiendo evadir de esta forma el uso de los armarios de control. Al mismo tiempo se abría el camino de la sustitución de un gran número de elementos electromecánicos por un solo elemento basado en circuitos de estado sólido, con lo que la fiabilidad aumentaba y la tasa de defectos y fallos disminuía.

Desde aquel entonces los PLC han ido evolucionando basándose tanto en microprocesadores como en la tecnología digital integrada. En la década de los setenta se implementó la comunicación mediante el bus de MODICON, MODBUS. De esta forma el PLC pudo comunicarse con otros PLC. En la década de los noventa se cambiaron los terminales de programación por un lenguaje simbólico y como intento de normalización del software de programación se estableció el estándar IEC 1131-3.

2 SISTEMA NORMALIZADO DE PROGRAMACIÓN DE AUTÓMATAS PROGRAMABLES

Se conoce como autómata programable a cualquier equipo electrónico capaz de controlar sistemas de tiempo real de forma secuencial. Por lo general, están basados en un microprocesador acompañado de los elementos necesarios para conseguir un determinado fin. Dependiendo del ámbito de aplicación al que se quieran destinar los controladores lógicos se pueden encontrar: micro-controladores (destinados a pequeñas aplicaciones) o controladores lógicos programables, también denominados PLC (destinados a entornos industriales).

El PLC es un elemento de control de procesos y de propósito general cuya implementación está destinada a la mayoría de situaciones en las que sea requerida una automatización. La captación de señales por parte del PLC es realizada por elementos sensibles al entorno del autómata y la realizarán elementos concretos llamados, de forma genérica, sensores. Tal ha sido la evolución de éstos, que ha permitido la realización de labores de control de forma cada vez más sofisticada.

Debido a este gran desarrollo de la microelectrónica, los autómatas programables están asociados a una gran cantidad de conceptos que dificultan el aprendizaje, lo que unido a la diversidad de lenguajes propietarios existentes hicieron que la Comisión Electrotécnica Internacional (IEC) elaborase el estándar internacional IEC 1131. El trabajo desarrollado en su tercera sección (1131-3) trata la estandarización en la programación del control industrial y puede descomponerse en dos partes:

- Elementos comunes
- Lenguajes de programación

2.1 Elementos comunes

2.1.1 Tipos de datos

Los datos forman la información básica con la que se realizan operaciones y se transmite la información. Se establecen los siguientes tipos de datos:

Denominación	Bits	Ejemplo	Descripción
BOOL	1	FALSE OR TRUE	Variable binaria o lógica
INT	16	-32768 … 32767	Entero con signo
UINT	16	0 … 65535	Entero sin signo
REAL	32	0.4560	Número real
BYTE	8	0 … 255	Conjunto de 8 bits
WORD	16	0 … 65535	Word
DWORD	32	0 … 2^{32}-1	Double Word
TIME		D#2002-02-02	Duración
DATE		T#2d2h2m2s2ms	Fecha
TIME_OF_DAY		TOD#12:02:22.02	Hora del día
DATE_AND_TIME		DT#2002-02-02-12:02:22.02	Fecha y hora
STRING		'CADENA'	Cadena de caracteres

Tabla 1- Principales tipos de datos del estándar IEC 1131-3. Fuente:[1]

2.1.2 Unidad de organización del programa (POU)

Cuando sea necesaria la resolución de aplicaciones sencillas, el proyecto desarrollado podrá y deberá constar únicamente de una tarea encargada de la ejecución cíclica del programa. Cuando se requiera la resolución de aplicaciones complejas lo adecuado será programar diversas tareas que se encarguen de la ejecución de las unidades de organización del programa (POU: Program Organization Unit). Las POU son instrucciones o un conjunto de ellas, que están relacionadas entre sí y que proporcionan una funcionalidad determinada. De esta forma el proyecto se divide en partes fácilmente compresibles.

Existen tres tipos de unidades de organización del programa:

Subprogramas

Dependiendo del proceso requerido y las secuencias de control deseadas, se podrán utilizar uno o varios programas pertenecientes al programa principal. Durante la ejecución de cada subprograma, el programa principal se mantendrá en suspensión.

Difieren de los bloques funcionales en el hecho de que pueden suministrar y recibir variables y valores, de la forma análoga a un programa escrito para un computador convencional, es decir, no constan de argumentos de entrada ni de salida y tienen como objetivos:

- Agrupar las instrucciones que se ejecuten varias veces a lo largo del programa
- Subdividir el programa en partes fácilmente comprensibles

Funciones

Se definen como una POU que al ser ejecutada proporciona tan solo un dato. Las funciones normalizadas no contienen información alguna de estado interno, o lo que es lo mismo, cada vez que se use una función con los mismos argumentos debe proporcionar siempre el mismo valor.

Además de las funciones preestablecidas, el programador puede declarar sus propias funciones y utilizarlas el número de veces que sea necesario.

Numéricas de una sola variable		**Aritméticas (de dos o más operandos)**	
ABS	Valor absoluto	ADD	Suma
SQRT	Raíz cuadrada	MUL	Multiplicación
LN	Logaritmo natural	**Aritméticas de dos operandos**	
LOG	Logaritmo en base 10	SUB	Resta
EXP	Exponencial natural	DIV	División
SIN	Seno en radianes	MOD	Módulo
COS	Coseno en radianes	EXPT	Elevación a exponente
TAN	Tangente en radianes	MOVE	Asignación
ASIN	Arcoseno		
ACOS	Arcocoseno		
ATAN	Arcotangente		
Decalaje		**Lógicas (Booleanas)**	
SHL	Desplazamiento hacia la derecha	AND	Operación lógica Y
SHR	Desplazamiento hacia la izquierda	OR	Operación lógica O
ROR	Rotación hacia la derecha	XOR	Operación O-exclusiva
ROL	Rotación hacia la izquierda	NOT	Negación
Selección		**Comparación**	
SEL	Selección	GT	Mayor
MAX	Máximo	GE	Mayor o igual
MIN	Mínimo	EQ	Igual
LIMIT	Limite	LE	Menor
MUX	Multiplexor	LT	Menor o igual
		NE	Desigual
Cadena de caracteres			
LEN	Longitud	INSERT	Inserción
LEFT	Caracteres a la izquierda	DELETE	Borrado
RIGHT	Caracteres a la derecha	REPLACE	Sustitución
MID	Caracteres intermedios	FIND	Búsqueda
CONCAT	Concatenación		
Conversión de tipo			
* TO **	Conversión de un tipo a otro		

Tabla 2 - Funciones establecidas por el estándar. Fuente: [1]

Bloques funcionales

Representan un algoritmo que, al ser ejecutado, proporciona una o más variables de salida. Se caracterizan por poseer variables de estado interno que pueden almacenar resultados parciales, por lo que a diferencia de las funciones, no generan siempre el mismo conjunto de valores de salida a partir de unos determinados valores de las entradas.

El programador podrá definir sus propios bloques funcionales y utilizarlos el número de veces que sea necesario.

BLOQUE FUNCIONAL	OPERADORES	DESCRIPCIÓN
SR	S1, R	Biestable RS de activación prioritaria
RS	S, R1	Biestable RS de desactivación prioritaria
R_TRIG	CLK	Convertidor de flanco ascendente en impulso
F_TRIG	CLK	Convertidor de flanco descendente en impulso
CTU	CU, R, PV	Contador ascendente
CTD	CD, LD, PV	Contador descendente
CTUD	CU, CD, R, LD, PV	Contador reversible
TP	IN, PT	Temporizador de impulso
TON	IN, PT	Temporizador de retardo a la conexión
TOF	IN, PT	Temporizador de retardo a la desconexión

Tabla 3 - Bloques funcionales predefinidos establecidos en el estándar. Fuente: [1]

2.1.3 Variables

Las variables conforman la información de los terminales de entrada y salida de un autómata programable así como la contenida en su memoria interna. La declaración de las variables permite su descripción mediante el nombre asignado.

Tipos de variables

Pueden ser distinguidas según sean definidas o no por el programador:

- Predefinidas: están establecidas en el lenguaje y podrán ser utilizadas en cualquier parte del programa.

- No predefinidas: podrán ser accesibles desde cualquier parte del programa o tan sólo desde la unidad de organización desde la que es definida, dependiendo del carácter global o local establecido en su declaración.

Palabra clave	Utilización de la variable
VAR	Variable local interna a la unidad de organización en la que se declara
VAR_INPUT	Suministrada externamente. No modificable dentro de la unidad de organización
VAR_OUTPUT	Suministrada por la unidad de organización
VAR_IN_OUT	Suministrada externamente. Modificable dentro de la unidad de organización
VAR_EXTERNAL	Suministrada externamente. Es modificable dentro de la unidad de organización
VAR_GLOBAL	Variable global
VAR_ACCESS	Variable accesible a través de una vía de comunicación
RETAIN	Variable no volátil o retentiva (no pierde información que contiene al dejar de aplicar la tensión de alimentación)
CONSTANT	Constante (no puede ser modificada)
AT	Asignación local

Tabla 4 - Palabras clave para la declaración de variables. Fuente: [1]

2.1.4 Identificación de variables

Con el fin de acceder directamente a las áreas de datos de los procesadores del PLC y de sus módulos de entrada/salida en el programa, IEC 61131-3 ofrece al programador dos posibilidades:

- Representar las variables directamente (Tabla 5)

  - Variables de entrada: el término "%In" (Input) representa a una variable lógica asociada con un número "n" correspondiente a su posición en el conector de entrada.

  - Variables de salida externas: el término "%Qn" (Output) representa a una variable lógica asociada con un número "n" correspondiente a su posición en el conector de salida.

  - Variables de salida internas: el término "%Mn" (Mark) representa una variable lógica interna (elemento de memoria) que tendrá asociado un número "n" correspondiente a su situación en la memoria del autómata. El número "n" distinguirá unas variables internas de otras y podrá estar formado por un número decimal o dos separados por un punto.

- Uso de variables simbólicas

Direct PLC addresses				Explanations
%				introductory character
	I Q M			input out flag/memory
		None X B W D L *		 bit bit (optional) byte word double word long word memory location, not (yet) defined
			v, w, x, y, z	multi-digit hierarchical address, increasing in significance from right to left. The number and interpretation of the places are dependent on the manufacturer, e.g.: z - bit, y - word, x - module, w - bus, v - PLC

Tabla 5 – Estructura de la representación directa de las variables. [2]

2.2 Lenguajes de programación

La selección del lenguaje de programación depende de factores diversos tales como la experiencia del programador, la aplicación concreta, el nivel de definición (descripción) de la aplicación, la estructura del sistema de control, el grado de comunicación con otros departamentos de la empresa…

Los distintos lenguajes establecidos permiten ingresar un programa de control en la memoria del PLC usando las sintaxis que en ellos se define:

- Lenguajes literales: las instrucciones están formadas por letras, números y símbolos.
    - Lista de instrucciones (IL: Instruction List). Consiste en la elaboración de una lista de instrucciones asociadas a símbolos y a su combinación (en un circuito eléctrico), a contactos.
    - Texto estructurado (ST: Structured Text). Consiste en una serie de instrucciones que pueden ser ejecutadas de forma condicionada o en bucles secuenciales.
- Lenguajes gráficos: las instrucciones están formadas por figuras geométricas.
    - Esquema de contactos (LD: Ladder Diagram). Consiste en la representación de variables lógicas mediante relés y de los contactos asociados a ellos.
    - Diagrama funcional de secuencias (SFC: Sequential Function Character). Consiste en gráficos de etapas y transiciones que permitirán describir el desarrollo de distintas acciones en el tiempo dentro de un programa. Surge como consecuencia directa del

lenguaje GRAFCET (Grafo de control etapa-transición) desarrollado por la Asociación Francesa para la Cibernética Económica y Técnica (AFCET).

- Diagrama de funciones (FBD: Function Block Diagram). Consiste en una lista de redes en la que cada una de ellas contiene una estructura que representa una expresión lógica o aritmética.

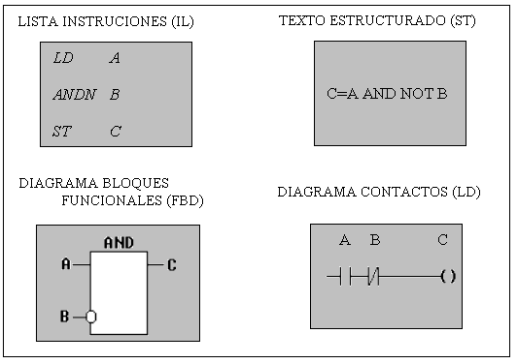


Figura 1 - Tipos de lenguaje. Fuente: [18]

2.3 Información general relativa a la conformidad con la norma IEC 61131- 3[1]

"*La norma IEC 61131-3 especifica la sintaxis y semántica de un conjunto unificado de lenguajes de programación para controladores programables. Éstos controladores están compuestos por dos lenguajes textuales, IL y ST, y dos lenguajes gráficos, LD y FBD. Además, los elementos del lenguaje de gráfica de función secuencial se definen para estructurar la organización interna de los programas de controladores programables y los bloques de función. También se definen los elementos de configuración, que admiten la instalación de programas de controladores programables en los sistemas de estos últimos.*

***NOTA:** Unity Pro utiliza las siglas inglesas para los lenguajes de programación. Asimismo, se definen las funciones que facilitan la comunicación entre controladores programables y otros componentes de los sistemas automatizados.*

2.3.1 Conformidad de Unity Pro con la Norma IEC 61131-3

La versión actual del sistema de programación Unity Pro admite un subconjunto compatible de los elementos de lenguaje definidos en la norma.

En este caso, compatible quiere decir lo siguiente:

- *La norma permite al encargado de implementar un sistema de programación IEC elegir o cerrar las funciones de lenguaje específicas o incluso completar lenguajes fuera de las tablas de función que forman parte inherente de las especificaciones; un sistema que solicite conformidad con la norma debe ejecutar las funciones elegidas de acuerdo con las especificaciones de la norma.*

- *Además, la norma permite al encargado mencionado con anterioridad utilizar los elementos del lenguaje de programación definido en un entorno de programación interactivo. Debido a que la norma afirma explícitamente que la especificación de dichos entornos no está dentro de su competencia, dicho encargado posee cierta libertad para proporcionar una presentación optimizada y procedimientos de manipulación para elementos de lenguaje específicos en beneficio del usuario.*

- *Unity Pro utiliza esta libertad mediante, por ejemplo, la introducción de la noción "Proyecto" para la manipulación combinada de los elementos de lenguaje IEC "Configuración" y "Recurso". Igualmente, hace uso de su libertad, por ejemplo, en los mecanismos proporcionados para la manipulación de declaraciones de variable o las instanciaciones de bloques de funciones.*

[1] Fuente: [12]

2.3.2 Extensiones de la norma IEC 61131-3

Además de las funciones IEC estándar enumeradas en las tablas de conformidad con las normas IEC, el entorno de programación de Unity Pro heredó un número de funciones del entorno de programación de PL7. Estas extensiones se proporcionan de forma opcional; pueden ser verificadas o no en el correspondiente cuadro de diálogo de opciones. El cuadro de diálogo y las funciones se describen en detalle en un capítulo de la ayuda online titulado Datos y lenguajes. En el cuadro de diálogo de opciones no está incluida otra extensión, que se heredó de los entornos de programación de PL7 y Concept: Unity Pro proporciona las construcciones de dicha sección en todos los lenguajes de programación, lo que permite la subdivisión de una unidad de organización de programa. Esta construcción presenta la posibilidad de combinar varios lenguajes (por ejemplo, secciones FBD, secciones SFC) en un cuerpo POU, función que, en caso de utilizarse con este fin, constituye una extensión de la sintaxis de IEC. Un cuerpo POU compatible debería contener una única sección. Las secciones no crean un campo de aplicación de nombre diferente. El campo de aplicación de nombre para todos los elementos de lenguaje es el POU.

Propósito de las secciones

Las secciones tienen diferentes propósitos:

- *Las secciones permiten subdividir cuerpos POU grandes de acuerdo con aspectos funcionales: el usuario tiene la posibilidad de subdividir el cuerpo POU en partes con funcionalidad significativa. La lista de secciones representa un tipo de tabla de contenidos funcional de un cuerpo POU amplio, que, de otro modo, estaría desestructurado.*

- *Las secciones permiten subdividir cuerpos POU grandes de acuerdo con aspectos gráficos: el usuario tiene la posibilidad de diseñar subestructuras de cuerpos POU grandes de acuerdo con una presentación gráfica. Puede crear secciones gráficas, grandes o pequeñas, según prefiera.*

- *La subdivisión de cuerpos POU grandes permite cambios online más rápidos: en Unity Pro, la sección se utiliza como unidad para cambio online. Si un cuerpo POU se modifica durante el tiempo de ejecución en ubicaciones diferentes, todas las secciones afectadas por los cambios se descargan automáticamente si se solicita explícitamente.*

- *Las secciones permiten volver a organizar el orden de ejecución de partes específicas y etiquetadas de un cuerpo POU: el nombre de la sección sirve como etiqueta de aquella parte del cuerpo que está contenida en la sección y, al ordenar estas etiquetas, se puede gestionar el orden de la ejecución de esas partes.*

- *Las secciones permiten utilizar distintos lenguajes de forma paralela en el mismo POU: esta función es una importante ampliación de la sintaxis de la norma IEC, que sólo permite la utilización de un único lenguaje IEC para un cuerpo POU. En un cuerpo compatible, SFC debe usarse para gestionar diferentes lenguajes dentro de un cuerpo (cada transición y acción debe formularse en su propio lenguaje).*

2.3.3 Diferencias de Unity respecto a las recomendaciones de IEC

Según IEC, el tipo de datos de la variable resultante no influye en el tipo de datos de la expresión resultante, y el tipo de datos de expresión se convierte en el tipo de datos resultante. Ejemplo:

i_DINT := REAL1+REAL2;

Equivalente utilizando la conversión de tipos explícita:

e_DINT := REAL_TO_DINT(REAL1+REAL2);

***NOTA:** La conversión de tipos implícita no está disponible para los lenguajes de programación SFC y LL984.*

2.3.4 Diferencias de Unity Pro

Unity Pro tiene estas excepciones respecto a las recomendaciones de IEC:

***1.** Si el tipo de datos de la variable resultante de una asignación es mayor que el tipo de expresión resultante, los parámetros de la expresión resultante se convierten en un tipo de parámetro de salida para evitar el desborde de la expresión.*

Ejemplo:

i_DINT := INT1 + INT2;

Equivalente utilizando la conversión de tipos explícita:

e_DINT := INT_TO_DINT(INT1) + INT_TO_DINT(INT2);

***2.** Unity Pro utiliza una conversión de tipos implícita para funciones genéricas; el tipo de datos de la variable resultante influye en el tipo de datos de la expresión resultante (función genérica).*

Ejemplo:

i_DINT := ADD (IN1 := INT1, IN2 := INT2);

Equivalente utilizando la conversión de tipos explícita:

e_DINT := ADD (IN1 := INT_TO_DINT(INT1), IN2 := INT_TO_DINT(INT2));

Los parámetros de salida genéricos o los bloques de funciones no influyen en el tipo de datos de la expresión resultante. Las conversiones de tipos de parámetros no coincidentes se ejecutan antes de llamar al cuerpo de FFB y la conversión de tipos de los parámetros de salida se ejecuta tras la llamada. Las conversiones de tipo implícitas, al contrario que las conversiones de tipo explícitas, solo se ejecutan cuando se llama al cuerpo de FFB.

Ejemplo:

SAH_0 (IN := BYTE1, CLK := BOOL1, PV := WORD1, OUT => i_DINT); “

3 BLOQUES FUNCIONALES EN UNITY PRO

En los bloques funcionales se implementan algoritmos en base a elementos definidos en el programa. Tienen como objetivo principal facilitar al usuario el empaquetado de algoritmos utilizados frecuentemente. Al igual que las funciones, los bloques funcionales pueden estar predefinidos o ser definidos por el programador.

Para la utilización de un bloque funcional será necesaria la definición de una variable que indique el tipo de bloque que se implementará así como la ejecución de las siguientes acciones:

- Asignación de variables del programa a las entradas del bloque
- Llamada condicional o incondicional al bloque
- Asignación de salidas del bloque a variables del programa

El software Unity Pro permite que el usuario pueda crear distintos tipos de bloques funcionales usando los lenguajes de automatismos y los designa formalmente como bloques funcionales derivados (DFB), también conocidos como bloques de funciones del usuario ya que el código de éstos se realiza con la finalidad de responder a características específicas del programador. La programación de bloques funcionales de usuario en el software para autómatas de Schneider está limitada al uso de los siguientes lenguajes de automatismos: LD, ST, IL y FBD.

3.1 FFB

Los diversos tipos de bloques funcionales que existen en Unity Pro pueden ser representados mediante las siglas FFB. Estas siglas engloban a:

- EF (Elementary Function): Función elemental
- EFB (Elementary Function Block): Bloque de funciones elementales
- **DFB (Derived Function Block): Bloque de funciones derivado**
- Procedimientos: A diferencia de los EF, constan de múltiples salidas

Cada FFB estará formado, por los campos ilustrados en la Figura 2:

❖ El nombre del FFB, el cual debe describir el tipo de operación que realiza.

❖ Los operandos necesarios para su funcionamiento. Existiendo dos tipos de operandos: los formales, encargados de transmitir datos desde o hacia un FFB durante el tiempo de ejecución del programa y los reales, que estarán vinculados con los parámetros formales. El tipo de dato de los parámetros formales relacionados con los reales deberá ser el mismo.

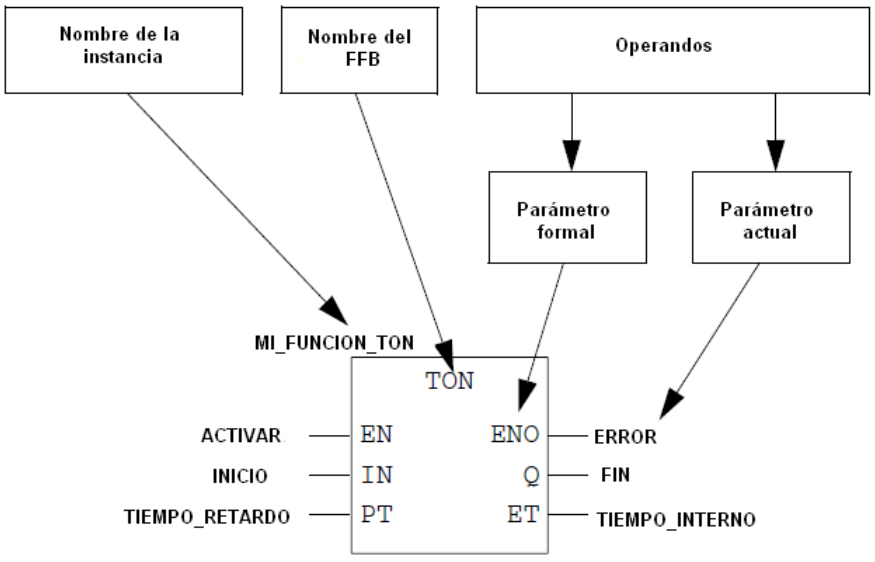


Figura 2 - Campos de un FFB. Fuente: [16]

3.2 Características generales

Los bloques funcionales de usuario pueden estar formados por una o varias secciones. Las secciones son entidades autónomas de programación que se ejecutarán en el orden en que aparecen en la vista estructural de la ventana del navegador.

La Tabla 6 muestra las características generales de una sección y a la hora de interpretarla se deberá tener en cuenta la imposibilidad de creación de secciones en lenguaje SFC en DFB.

Características	**Descripción**
Nombre	Limitado a 32 caracteres, permitidos los acentos pero no los espacios
Lenguaje	LD, FBD, IL, ST
Tarea o tratamiento	Maestra, rápida, auxiliares, de sucesos
Condición (opcional)	Variable tipo BOOL o EBOOL condiciona la ejecución de la sección
Comentario	Limitado a 256 caracteres
Protección	Protección contra la escritura y lectura/escritura

Tabla 6 - Características generales de una sección en Unity Pro. Fuente: [3]

Aunque en algunos manuales de referencia es posible encontrar la siguiente cita[2], es posible comprobar tanto que en la versión de la suite informática utilizada (4.0) no se encuentra disponible dicha opción como que no es necesario el ajuste de ningún parámetro para que el correcto funcionamiento de varias secciones sea posible en un DFB.

> *"Para los DFB, es el usuario quien los escribe en lenguaje de automatismo (literal estructurado, lista de instrucciones, lenguaje de contactos, lenguaje de bloques funcionales) y está estructurado en una sola sección si la opción IEC está activa, o bien puede estructurarse en varias secciones si esta opción está inactiva"*

La creación de un DFB supone únicamente la declaración de un tipo de bloque compatible con Unity Pro. Para la utilización de éste en el programa se debe realizar una invocación. A las invocaciones se les denomina instancias, las cuales se identifican mediante un nombre y poseen los tipos de datos del DFB correspondiente. En la Tabla 7 se muestran las características generales de un DFB.

Nombre	32 caracteres
Comentario	1024 caracteres
Datos de entradas	32 máximo
Datos de entradas/salidas	32 máximo
Datos de salidas	32 máximo
Número de interfaces (IN + OUT +IN/OUT)	32 máximo
Datos públicos	Sin límites
Datos privados	Sin límites
Lenguaje de programación	LD, ST, IL, FBD

Tabla 7 - Características generales DFB. Fuente: [3]

[2] Fuente: [15]

3.2.1 Parámetros

Los bloques funcionales necesitan, para poder funcionar correctamente, diversos argumentos mediante los que recibir y entregar valores en un programa o en un subprograma.

- **Entradas**: Permiten pasar valores del programa de aplicación al DFB.

- **Salidas**: Permiten pasar valores internos del DFB al programa de aplicación.

- **Entradas/Salidas**: Permiten pasar los datos del programa de aplicación al DFB, que los puede modificar y pasarlos de nuevo al programa de aplicación.

- **EN y ENO**: EN es un parámetro de entrada que permite, cuando está a nivel alto, la ejecución de las instrucciones del DFB. ENO es un parámetro de salida que, cuando está a nivel alto, permite el condicionamiento de la ejecución en cascada de otros bloques.

Figura 3 – Estructura de un bloque funcional. Fuente: [12]

3.2.2 Variables

Conforman la información de los terminales de entrada y salida de un autómata programable además de la contenida en una posición de su memoria interna. Se distinguen, según la posibilidad de modificación de forma externa al bloque, dos tipos:

- Variables públicas: Variables internas del DFB que pueden ser usadas por el DFB, el programa de aplicación y por el usuario en el modo de ajuste.

- Variables privadas: Variables internas del DFB que sólo puede usar el DFB y por lo tanto no será posible su modificación a través del programa de aplicación. Suelen ser variables necesarias para la programación del bloque función pero de escaso interés para el usuario. Se puede acceder a este tipo de variables mediante una tabla de animación.

Para acceder a ambos tipos de variables, ya sea para visualizar o modificar variables públicas o para visualizar variables privadas, se deben implementar expresiones que sigan la siguiente estructura:

"Nombre_Instancia**.**Nombre_variable"

- "Nombre_Instancia" representa el nombre de la instancia del DFB (32 caracteres)
- "Nombre_variable" representa el nombre de la variable privada (8 caracteres)

3.2.3 Tipos de datos

Los datos forman la información básica con la que se realizan operaciones y se transmite la información. Unity realiza una extensión de algunos tipos de datos basándose en la normativa. En la Tabla 8 se muestran los tipos de datos predefinidos:

Denominación	**Bits**	**Ejemplo**	**Descripción**
BOOL	1	False or True	Variable binaria o lógica
EBOOL	2	False or True	Binaria o lógica con información de flancos
INT	16	-32768 … 32767	Entero con signo
DINT	32	-2147483648 … 2147483648	Entero doble con signo
UINT	16	0 … 65535	Entero sin signo
UDINT	32	0 … 4294967295	Entero doble sin signo
REAL	32	0.4560	Número real
BYTE	8	0 … 255	Conjunto de 8 bits
WORD	16	0 … 65535	Word
DWORD	32	0 … 2^{32}-1	Double Word
TIME	32	D#2002-02-02	Duración (UDINT)
DATE	32	T#2d2h2m2s2ms	Fecha
TIME_OF_DAY (TOD)	32	TOD#12:02:22.02	Hora del día
DATE_AND_TIME (DT)	64	DT#2002-02-02-12:02:22.02	Fecha y hora
STRING		'CADENA'	Cadena de caracteres

Tabla 8 - Tipos de datos predefinidos en Unity Pro.

También es posible la creación de tipos de datos derivados (DDT) cuyo formato dependerá de su contenido:

- Estructura: dato que contiene un conjunto de datos de distinto tipo. La creación de estructuras intercaladas es posible siempre que no se superen los ocho niveles. No es posible implementar estructuras recursivas.

- Tabla: elemento que contiene un conjunto de datos del mismo tipo.

3.3 DFB intercalado

Cuando un DFB se encuentra instanciado dentro de otro DFB se dice que está intercalado. Unity Pro cataloga los DFB intercalados en un nivel inferior como datos privados, por lo que su modificación a través del programa de aplicación no es posible. La expresión sintáctica a utilizar debe tener la siguiente estructura: "Nombre_Instancia**.**Nombre_Parametro"

- "Nombre_Instancia" representar el nombre de la instancia del DFB (32 caracteres)
- "Nombre_Parametro" representa el nombre del parámetro de salida (32 caracteres)

Cuando se creen DFB intercalados no se deben de sobrepasar los ocho niveles, incluidas las variables DDT. Además, es necesario respetar varias reglas cronológicas, las cuales dan forma al siguiente procedimiento:

- Crear el DFB del último nivel (n)
- Crear el DFB del nivel (n-1)
- Para este DFB, crear una variable privada que contenga como el DFB del nivel n. Al instanciar un DFB dentro de otro esta variable se crea automáticamente.
- Crear el DFB de nivel (n-2)
- Para este DFB, crear una variable privada que contenga el DFB de nivel (n-1)
- Repetir estas acciones sin sobrepasar los ocho niveles

3.4

3.5 Creación de DFB

Ante las ventajas y características presentes en los DFB surge la necesidad de su utilización. A continuación se analiza el proceso de declaración que permite conformar los bloques de funciones de usuario.

La creación de bloques derivados será llevada a cabo mediante el editor de datos, accesible desde la Vista estructural -> Variables e instancias FB y desde Herramientas. Este menú consta de los siguientes submenús:

- Variables: Definición de todas las variables del proyecto
- Tipos de DDT: Definición de los tipos de datos derivados
- Bloques de funciones: Contiene las instancias de DFB
- **Tipos de DFB**: Definición de los tipos de DFB

El usuario deberá seleccionar la pestaña correspondiente a los tipos de DFB. La declaración comienza desde el editor de datos con la introducción del nombre del bloque, Figura 4.

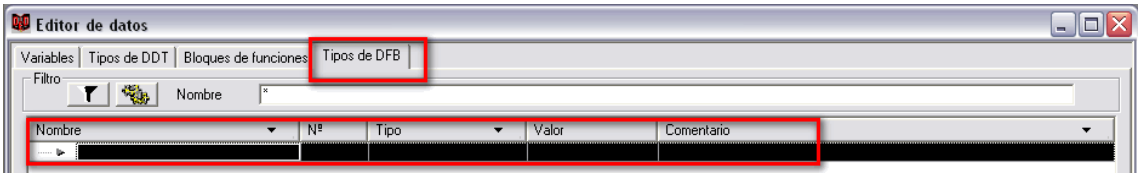


Figura 4 - Editor de datos: Tipos de DFB

Una vez nombrado, un icono de obras indicará que el DFB está en construcción (Figura 5). Para que se pueda considerar construido se deberá generar el proyecto tras asignar los parámetros de entrada, salida y entrada/salida del bloque. Desde el editor, se podrán definir las variables públicas y privadas aunque también será posible definirlas a medida que se implemente el código en las diferentes secciones.

El establecimiento de valores iniciales en distintos argumentos se realizará desde el mismo menú. Se debe prestar especial atención a la asignación de valores iniciales a argumentos que, una vez instanciados, pueden ser direccionados.

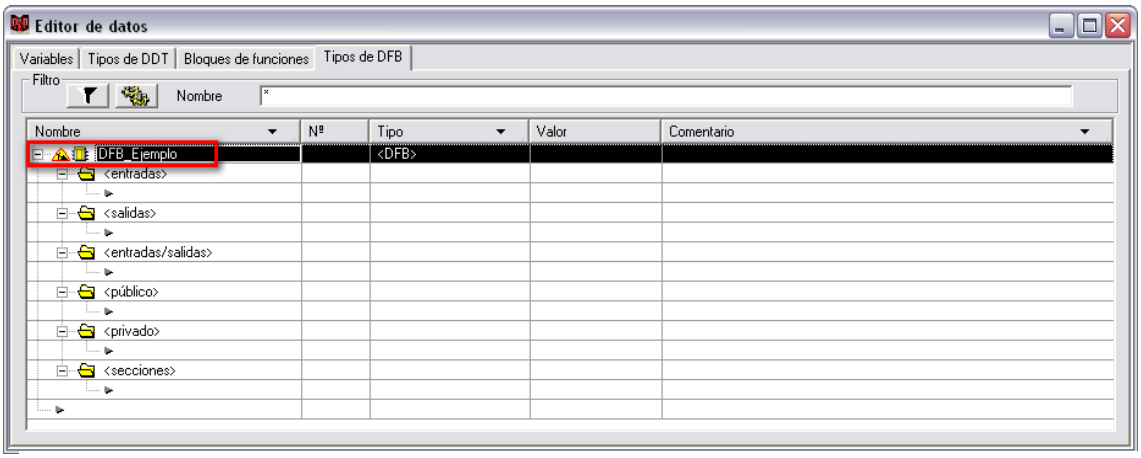


Figura 5 – Editor de datos: DFB en construcción

Las secciones implementarán el código del programa y deben ser definidas mediante el editor de datos (Figura 7) o desde el apartado secciones de cada DFB (Figura 6).

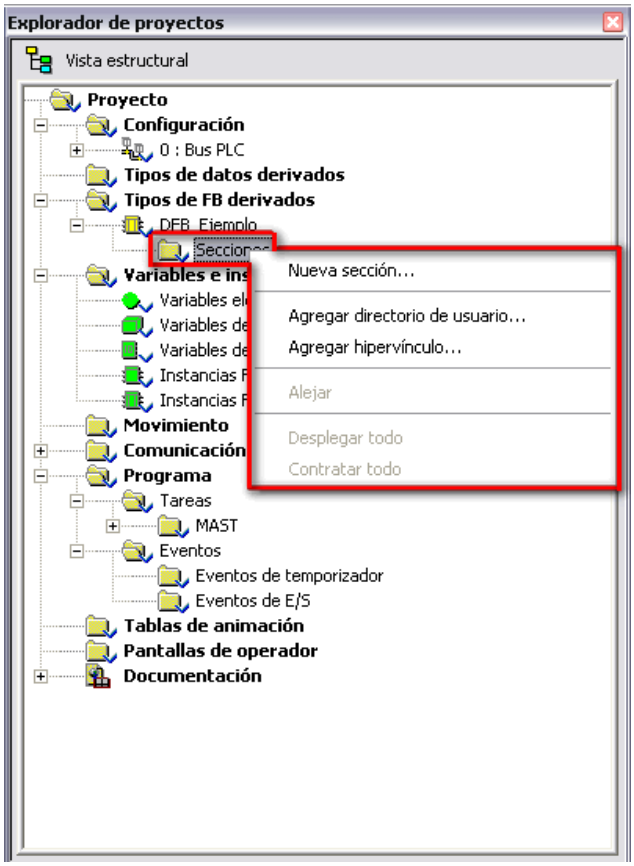


Figura 6 – Explorador de proyectos: Nueva sección

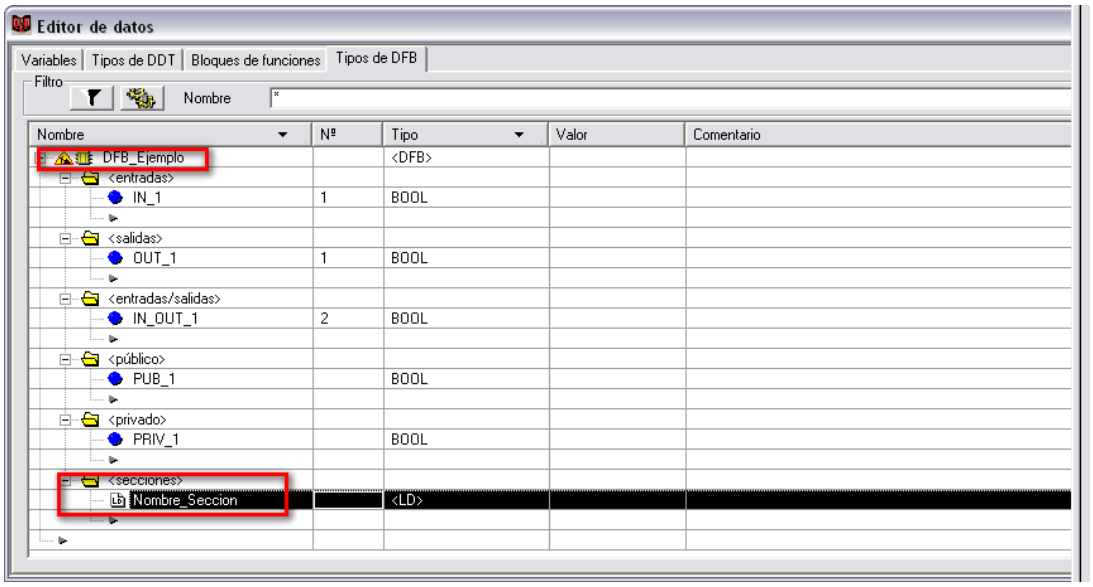


Figura 7 – Editor de datos: Creación de sección

La creación de una sección está condicionada a la elección de un nombre, de un lenguaje (Figura 8) y de un posible condicionamiento (booleano) para su activación (Figura 9).

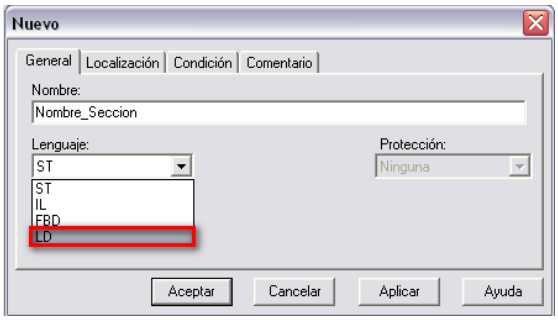


Figura 8 – Creación de sección: Elección lenguaje

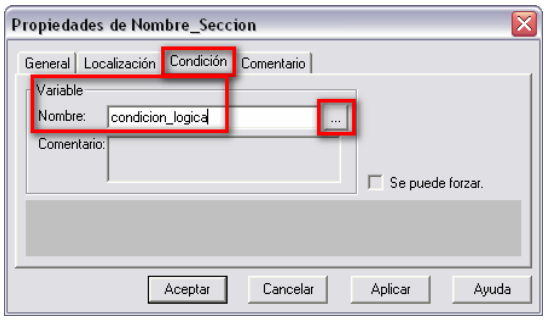


Figura 9 - Propiedades de sección: Condición

Tras realizar los pasos anteriores el usuario puede observar que se sigue mostrando el icono de DFB en construcción. Esto es debido a que aún habiendo definido tanto el código como los parámetros necesarios para su funcionamiento, será necesario analizar y generar los cambios en el proyecto, con el fin de que el compilador detecte errores y/o genere advertencias de uso.

- Analizar: Analiza el proyecto y descubre errores
- Generar cambios: Analiza y genera las partes modificadas del proyecto
- Regenerar todo el proyecto: Analiza y regenera todo el proyecto. Las variables internas tan sólo de reiniciarán al usar el comando de Regenerar todos los proyectos.

El comando Generar cambios puede ejecutarse sólo si el comando Regenerar todos los proyectos se ha ejecutado previamente al menos una vez.

Cuando un proyecto es analizado (Figura 10) y no genera errores, visualmente, se genera una señal azul de verificación (Figura 11).

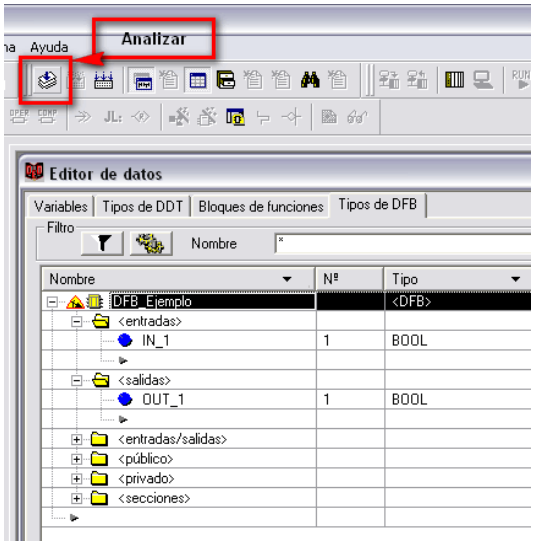


Figura 10 - Analizar programa

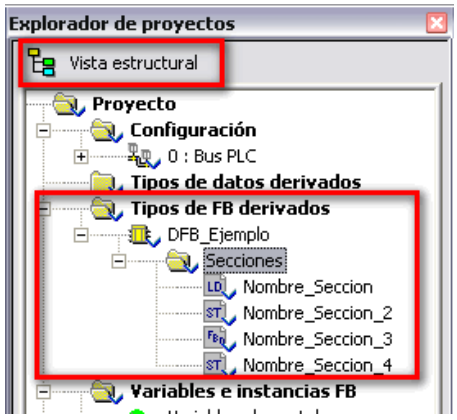


Figura 11 – Explorador proyectos: Secciones de DFB analizado

Al generarse o regenerarse un proyecto correctamente, la simbología adjunta al icono del DFB desparece (Figura 12).

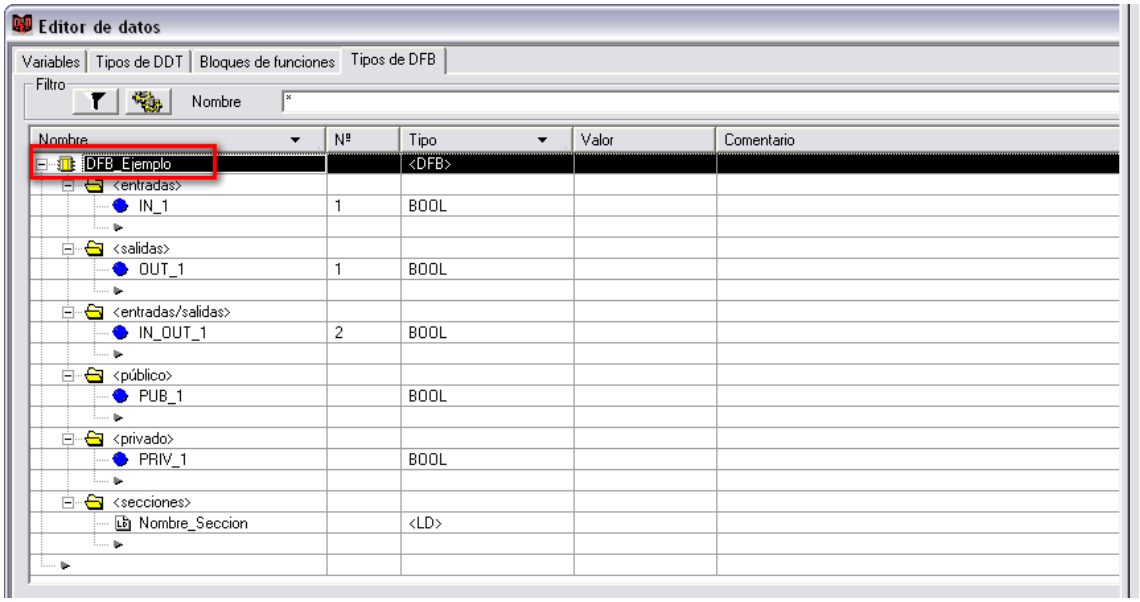


Figura 12 – Editor de datos: DFB generado correctamente

Mediante el menú contextual Figura 13, accesible desde Figura 12, es posible la eliminación de las instancias de datos privados que no hayan sido utilizados. Es recomendable el uso de esta opción, una vez se haya comprobado que el bloque funcional cumpla su cometido. También es posible tramitar la supresión de los tipos de DFB que no hayan sido instanciados en el programa.

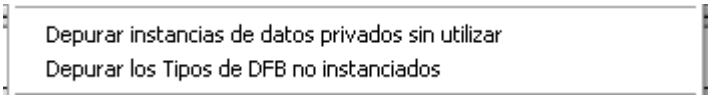


Figura 13 – Eliminación de instancias y tipos no usados en DFB

3.6 Creación de instancias de DFB

Definido y generado correctamente el tipo de DFB, se puede invocar desde cualquier sección del programa o en otras definiciones de DFB. Las invocaciones se realizan mediante la creación de instancias, las cuales se pueden utilizar en todas las tareas del programa de aplicación siguiendo la estructura de:

- un bloque de función estándar en LD o FBD
- una función elemental en ST o IL

Será posible la creación de instancias de DFB mediante el acceso al menú edición y/o la selección de alguna de las siguientes opciones siempre y cuando se proceda a ello desde alguna sección del programa o desde una sección de un bloque funcional derivado (cuando se desee intercalar bloques). Este proceso se realiza indistintamente de si se está intercalando un bloque o de si se implementa en una sección maestra (Figura 14).

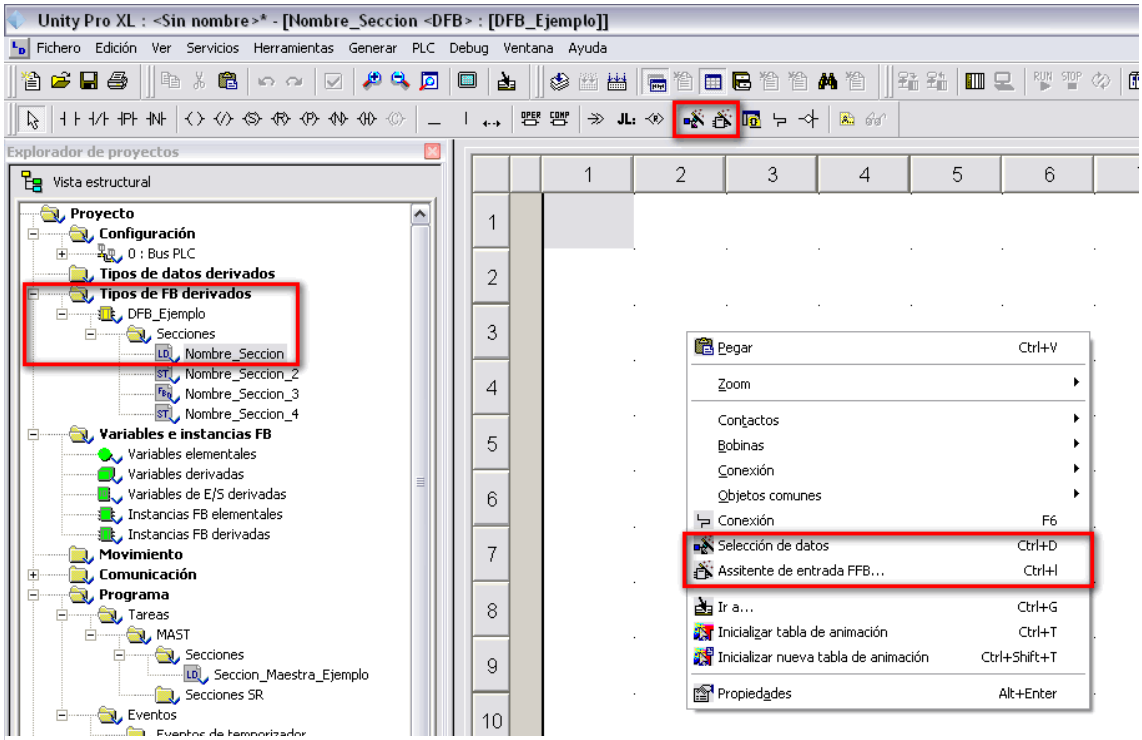


Figura 14 - Instanciado de Bloque DFB

Las instancias de DFB derivados pueden tener asociados valores iniciales. En el caso de que a una variable se le otorgue valores iniciales desde la definición del tipo de DFB y desde una instancia del tipo de DFB se asigna como valor de referencia el de la instancia.

3.6.1 Selección de datos

Mediante el comando de selección de datos se accede directamente al menú de selección de FFB. En este menú se muestran las librerías del programa (Figura 16) así como de los DFB que hayan sido creados en la aplicación (Figura 15) aunque no estén ubicados en la librería.

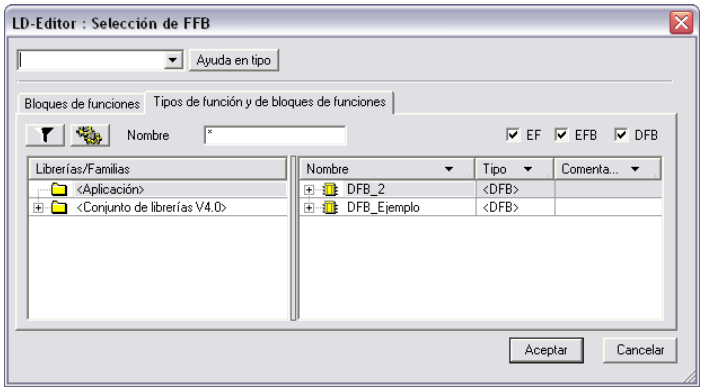


Figura 15 – Instanciado de DFB no ubicado

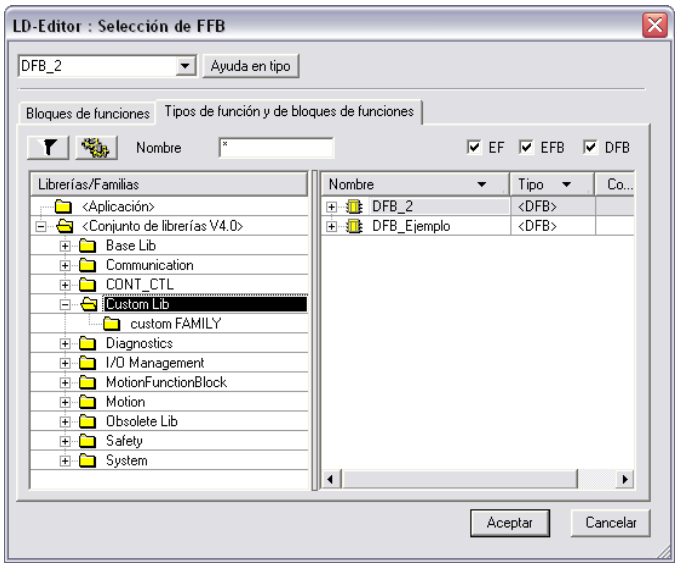


Figura 16 - Instanciado de DFB ubicado

3.6.2 Asistente de entrada FFB

El asistente de entrada de funciones permite, a través de la zona indicada en la Figura 17, el acceso al menú de selección de FFB comentado en el apartado anterior. La diferencia radica principalmente en la posibilidad de otorgar un nombre a la instancia que se esté creando además de una visualización del prototipo de la instancia.

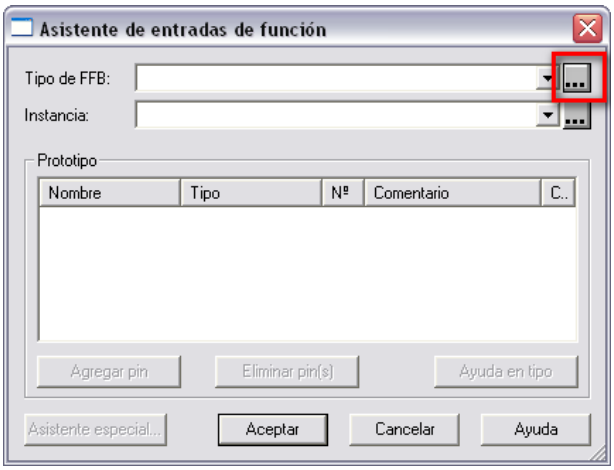


Figura 17 – Asistente de entrada de función

3.7 Protección de DFB

Los bloques funcionales derivados podrán asegurarse ante posibles modificaciones de usuarios no autorizados según distintos niveles de protección (Figura 18):

- **Sólo lectura**: Los directorios de parámetros de los tipos de DFB están en formato de sólo lectura.
- **Protección de la versión**: El tipo de DFB no está protegido, exceptuando el número de versión de DFB.
- **Sin lectura y escritura**: Los directorios de parámetros de los tipos de DFB privado y secciones no se muestran. Puede accederse a todos los demás directorios de parámetros de los tipos de DFB desde el editor de datos en formato de sólo lectura.
- **Sin protección**: El tipo de DFB no está protegido.

Si un DFB utiliza un DDT, se puede modificar el tipo de DDT incluso si el DFB está protegido.

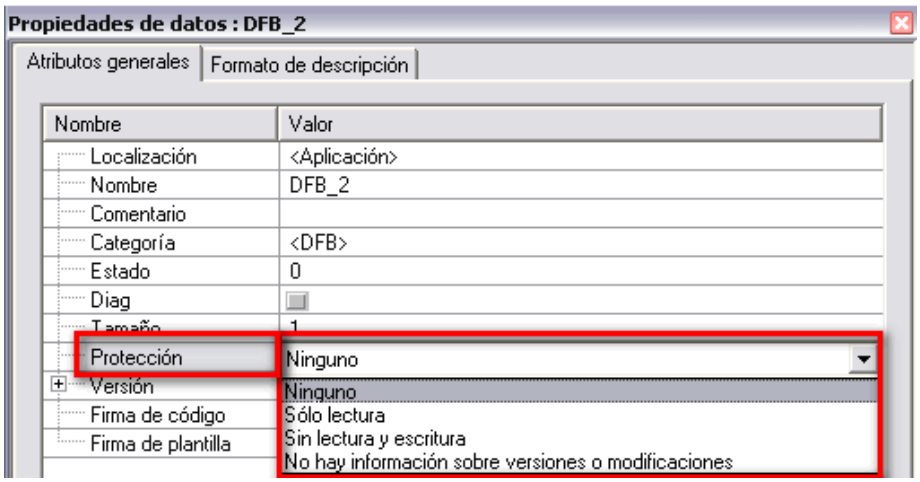


Figura 18 – Protección de DFB

3.7.1 Modificación de nivel de protección

Para cambiar el nivel de protección de un tipo de DFB, se deben seguir lo siguientes pasos:

- Seleccionar el DFB que se desea proteger
- Hacer click derecho sobre éste o acceder a Edición -> Propiedades (Figura 19)
- Introducir una contraseña de 8 caracteres máximo

Figura 19 – Modificar nivel de protección DFB

3.7.2 Modificación de la contraseña

Para cambiar la contraseña de un tipo de DFB, deberá seguir los pasos anteriores y seleccionar la nueva opción disponible, Figura 20.

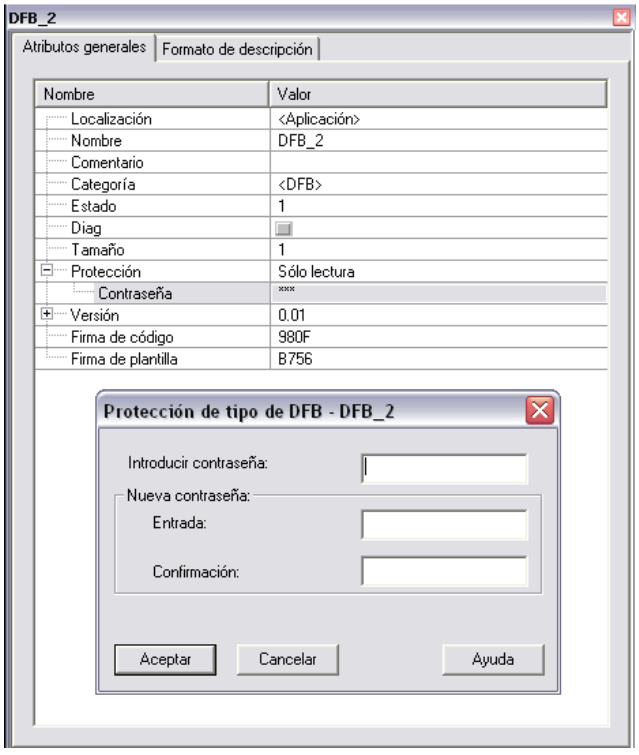


Figura 20 – Modicar contraseña DFB

3.8 Creación de ayuda en línea para DFB

Los DFB predefinidos tienen asociados un fichero de ayuda. Es posible vincular un archivo de ayuda HTML a los DFB creados y colocados en alguna biblioteca definida por el usuario.

Para ello, el fichero de ayuda HTML debe tener el mismo nombre que el DFB al que se desea vincular y tener una extensión *html. Los documentos correspondientes al fichero de ayuda tienen que ubicarse dentro de la carpeta de instalación de Unity Pro:

- Para Windows XP este fichero se debe ubicar en el directorio:

  C:\Documents and Settings\All Users\Application Data\Schneider Electric\Unity Pro\Custom Libset/Vx.x/Idioma[3]

- Para Windows Vista este fichero se debe ubicar en el directorio:

  C:\ProgramData\Schneider Electric\Unity Pro\CustomLibset/Vx.x/Idioma[4]

La creación de la ayuda en línea se realiza siguiendo los siguientes pasos:

- Crear un archivo HTML con cualquier tipo de editor HTML. Este archivo debe tener el mismo nombre que el DFB.
- Ubicar el archivo HTML en la carpeta de idioma correspondiente.
- Crear una carpeta adicional en el mismo directorio en que se sitúan las carpetas de idioma denominada HELP.
- Ubicar los ficheros referenciados en el HTML en la carpeta HELP.
- Instalar la nueva biblioteca en Unity Pro.

Como resultado todos los ficheros se copiarán en el directorio Libset y el fichero de ayuda HTML se iniciará al hacer clic en el botón de ayuda en tipo en el menú contextual del bloque.

[3] El idioma se deberá especificar como el nombre la carpeta según el idioma deseado: *ENG, FRE, GER, ITA, SPA* o *CHI*.
[4] El idioma se deberá especificar como el nombre la carpeta según el idioma deseado: *ENG, FRE, GER, ITA, SPA* o *CHI*.

4 EXPORTACIÓN/IMPORTACIÓN

La exportación e importación permite utilizar en proyectos diferentes los datos, secciones, pantallas de operador… creados por el usuario en el proyecto origen. Será posible acceder a ellas mediante las distintas carpetas accesibles desde la vista estructural del explorador de proyectos.

La función de exportación, permite copiar el programa de proyecto completo o parte del mismo a un fichero.

La función de importación permite recuperar el programa del proyecto completo o parte de él para utilizarlo en el proyecto.

Se pueden exportar los siguientes elementos:

- Proyectos completos
- Secciones de todos los lenguajes de programación
- Secciones de subrutinas de todos los lenguajes de programación
- **Bloques de función derivados (DFB)**
- Tipos de datos derivados (DDT)
- Tipos de datos derivados de dispositivo (DDT de dispositivo)
- Declaraciones de variables
- Pantalla del operador

4.1 Exportación de los tipos de DFB

La función de exportación genera un fichero que contiene datos no protegidos y referencias a datos protegidos. Es necesario establecer el nombre de este fichero, su ubicación (directorio) y la extensión (determinada por el elemento exportado).

La exportación de DFB implica la de los datos derivados presentes en ella. También será posible exportar un tipo de DFB protegido (en modalidad de lectura o escritura) y mantener el mismo tipo de protección.

El acceso a esta funcionalidad se realiza a través de la carpeta "Tipos de FB derivados" del explorador de proyectos. Según el elemento seleccionado en el árbol de directorios del explorador de proyectos, se podrán exportar:

- Todos los tipos de DFB del proyecto actual (incluso los que no se han utilizado)
- Un tipo de DFB.

1. Exportación de todos los DFB (Figura 21)

Al exportar todos los tipos de DFB del proyecto, el software genera un fichero, de extensión .XDB, que contendrá: los tipos de DFB intercalados no protegidos, los DDT utilizados y las referencias a los tipos de DFB protegidos.

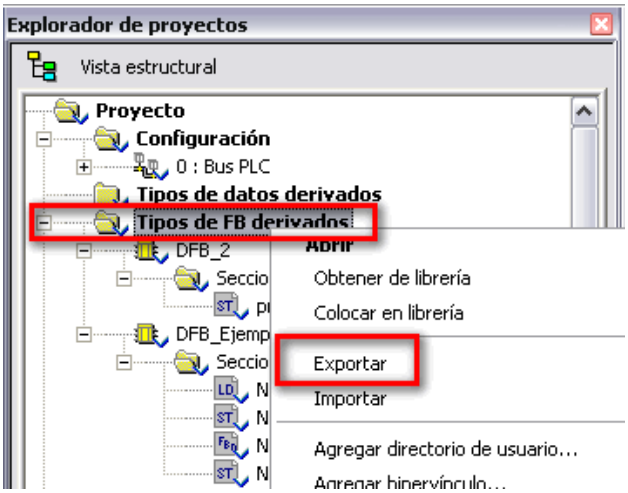


Figura 21 – Expotar todos DFB del programa

2. Exportación de cada DFB independientemente (Figura 22)

Al exportar un tipo de DFB, el software genera un fichero, de extensión .XDB, que contendrá toda la información no protegida: los tipos de DFB actuales, los tipos de DFB intercalados, los DDT utilizados y las referencias a los tipos de DFB protegidos.

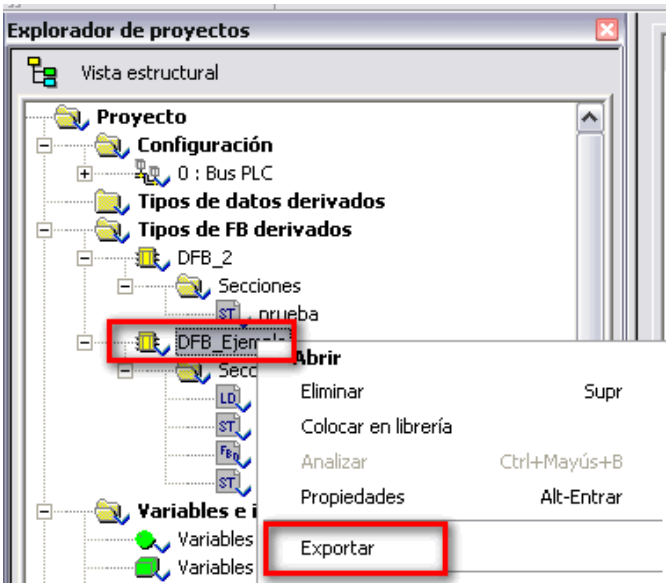


Figura 22 – Exportar DFB unitariamente

4.2 Importación de los tipos de DFB

Esta función posibilita la recuperación de bloques funcionales derivados previamente exportados. Su acceso se realiza desde la carpeta "Tipos de FB derivados" del explorador de proyectos y, de forma similar al caso de la exportación, se puede realizar la importación de uno o todos los tipos de DFB (Figura 23).

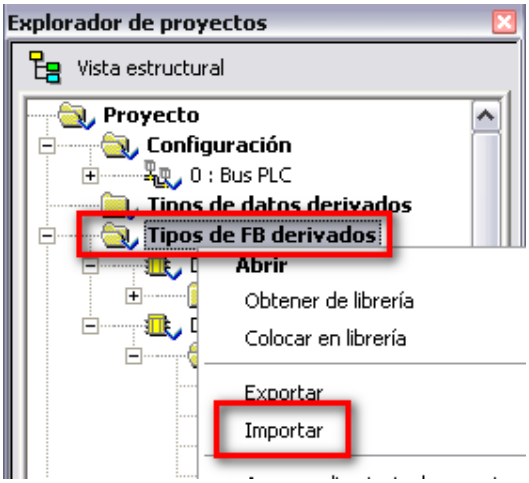


Figura 23 – Importar todos los DFB del programa

Es posible realizar la importación de un tipo de DFB protegido aunque si estaba protegido contra escritura, sólo se podrá acceder a los elementos del DFB en modalidad de lectura. Si estaba protegido contra lectura y escritura, no se podrá acceder a los elementos. Si se ha activado la Protección de la versión, se podrá acceder al número de versión sólo en modalidad de lectura y a los elementos del DFB en lectura y escritura. Si el tipo de DFB utiliza DFB o DDT intercalados, éstos también se importan.

Después de una importación, es necesaria la confirmación de los datos importados (análisis y generación) ya que ésta funciona como una entrada manual y no se confirma automáticamente.

4.3 Gestión de conflictos de importación DFB

Si tras realizarse la importación de un elemento hay otro en el proyecto cuyo nombre es el mismo, se debe adoptar una solución, mediante el gestor de problemas de importación (Figura 24):

- **Conservar**: el elemento presente en el proyecto se mantiene mientras que el elemento con el mismo nombre no se importa.

- **Reemplazar**: el elemento presente en el proyecto se reemplaza con el elemento importado con el mismo nombre.

- **Cambiar nombre**: se puede cambiar el nombre del nombre del elemento que se va a importar para solucionar el conflicto y disponer de ambos tipos.

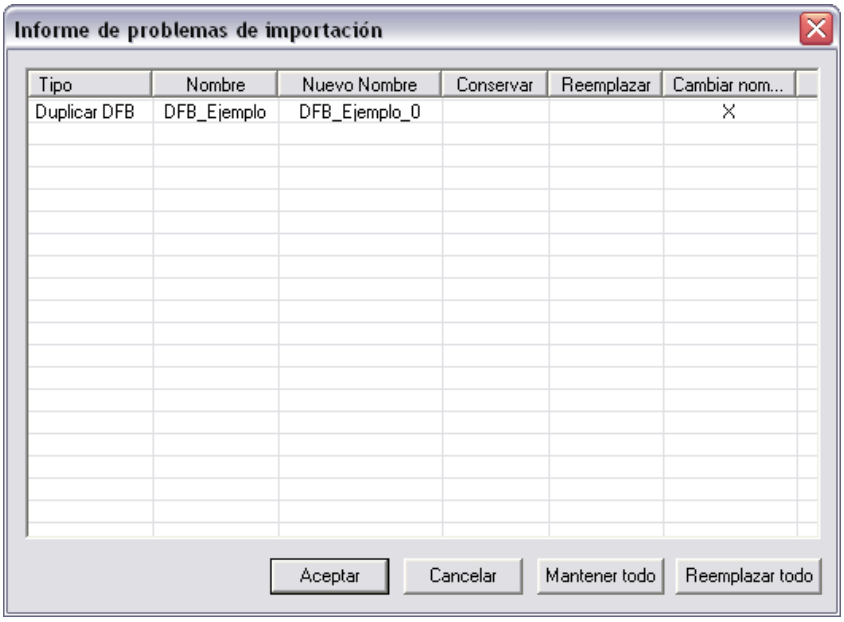


Figura 24 – Gestor de problemas de importación

4.4 Obtención de librería

Los DFB que hayan sido ubicados previamente en alguna librería se pueden obtener y colocar en el explorador de proyectos, para su posible modificación o análisis estructural.

Para realizar la importación de bloques de funciones de usuario es necesario el acceso al gestor de la librería de tipos (Figura 26) mediante, por ejemplo, el explorador de proyectos (Figura 25).

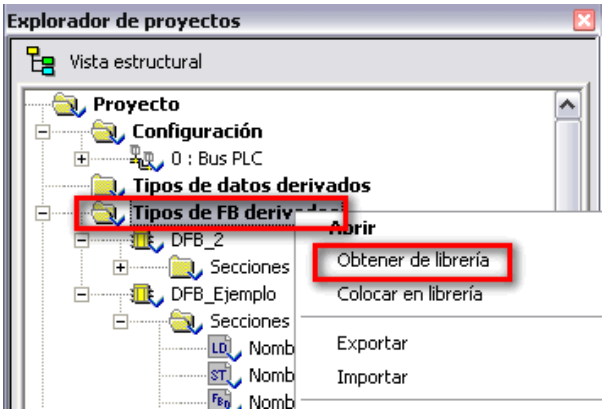


Figura 25 – Obtener DFB de librería

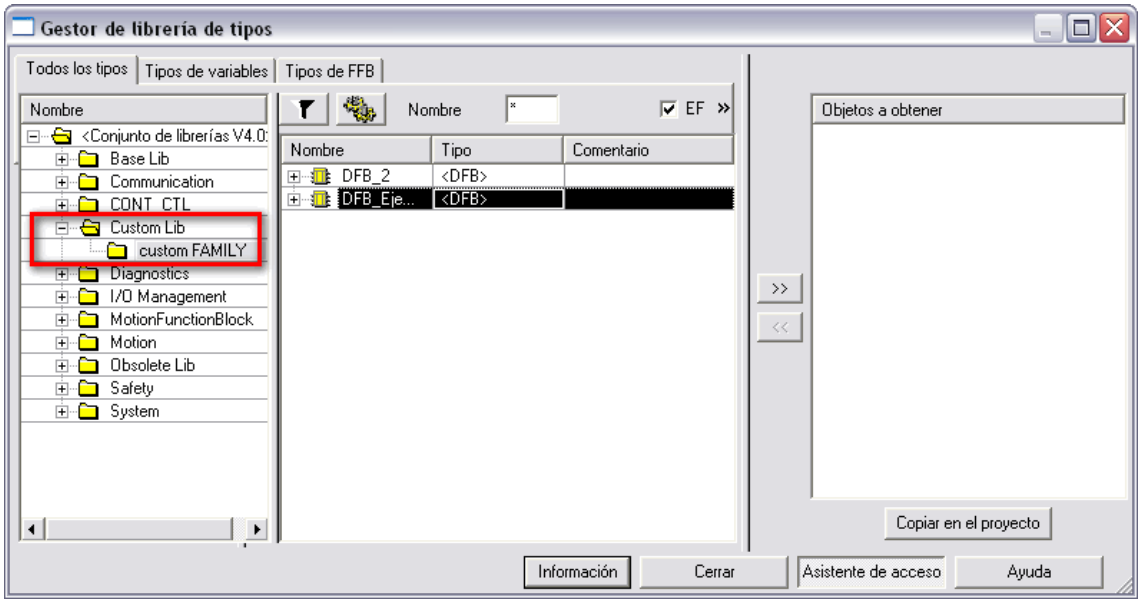


Figura 26 – Gestor de librería de tipos con asistente de acceso

En la Figura 27 se observan dos bloques funcionales ubicados en la librería personalizada “Custom Lib”. Para importar uno o varios se deben selecionar haciendo uso de las flechas de asignación, colocar en una lista con los objetos a obtener y activar el comando Copiar en el proyecto.

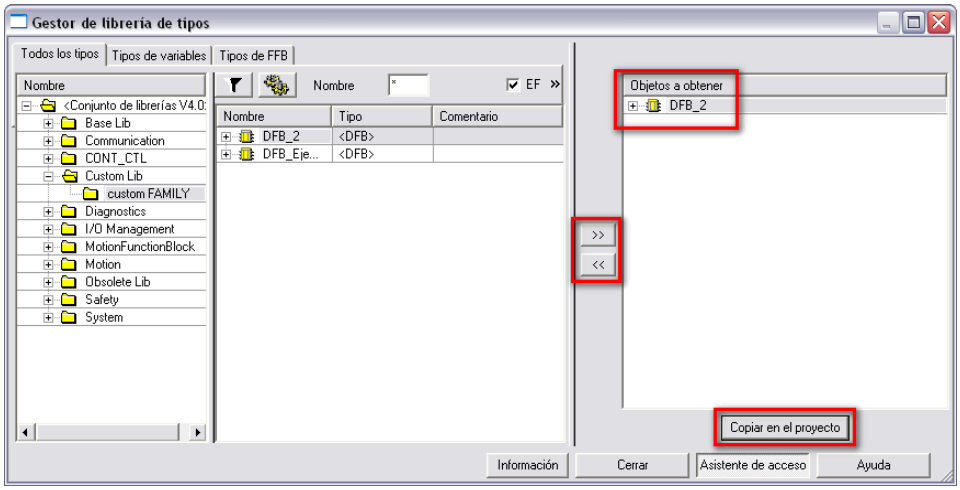


Figura 27 – Selección de objetos a obtener desde el gestor de librerías

4.5 Colocación de librería

La transferencia de los diferentes objetos presentes en un proyecto a una librería permite el uso de éstos en otro proyecto sin tener que importar o exportar entre aplicaciones. Se dispondrá de la posibilidad de inserción en la librería del programa siempre y cuando se haya realizado con éxito la creación del bloque función derivado. El acceso a los objetos transferidos a la librería es posible desde todos los proyectos.

Según el elemento seleccionado en el árbol de directorios del explorador de proyectos, se podrán colocar todos los DFB en la librería simultáneamente o unitariamente.

1. Colocación de todos los DFB en la librería (Figura 28)

Para transferir al unísono todos bloques funcionales derivados se debe acceder al menú contextual de los tipos de FB derivados desde el explorador de proyectos y seleccionar la ubicación de la librería donde ser almacenados (Figura 29).

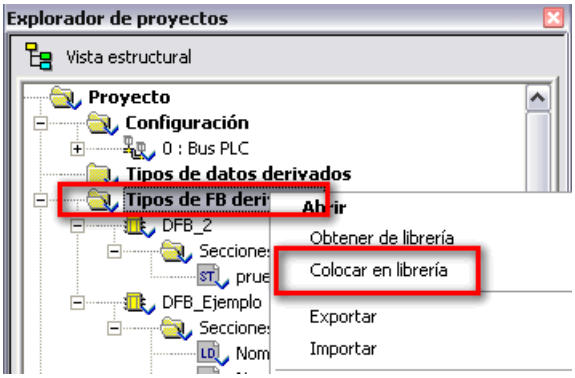


Figura 29 – Selección librería destino de DFB

Figura 28 - Colocar todos los DFB en librería

2. Colocación de DFB unitariamente en la librería

A diferencia del caso anterior se seleccionarán de forma individual los bloques a transferir (Figura 30).

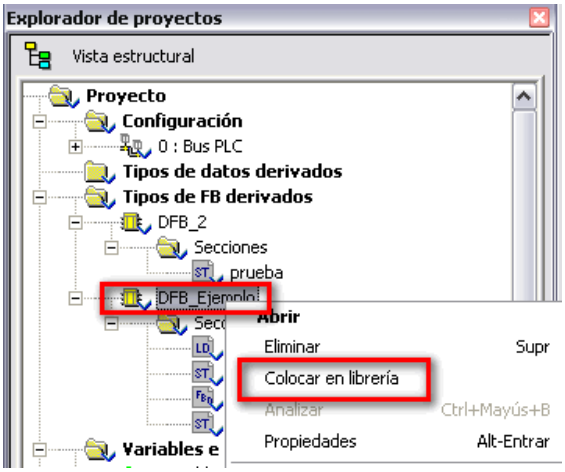


Figura 30 – Colocar DFB en librería

La colocación de tipos en la librería dispone de comprobación de sobre-escritura de tipos del mismo nombre (Figura 31)

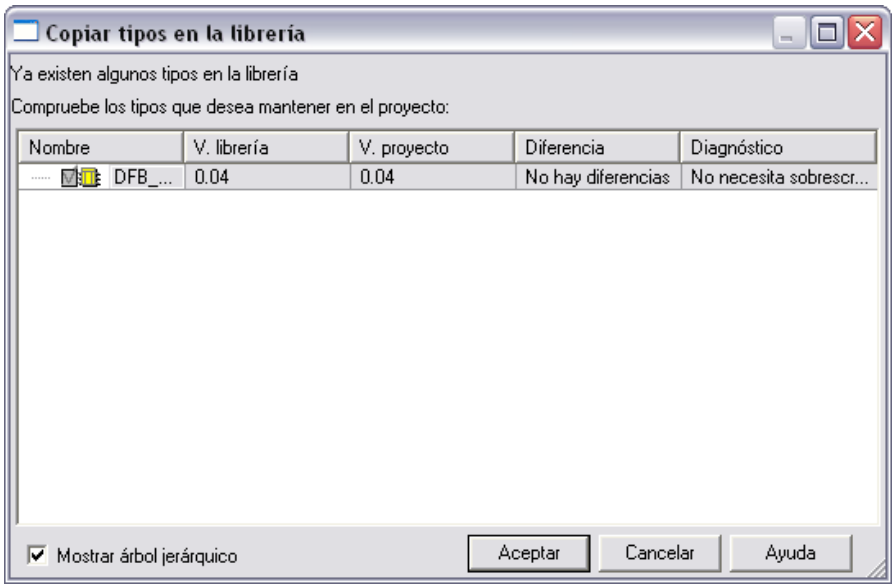


Figura 31 – Comprobación de sobre-escritura de tipos DFB

4.6 Eliminación de librería

En el caso de que el usuario desee suprimir un DFB previamente colocado en una librería, se deb acceder al gestor de la librería de tipos, seleccionar los DFB a eliminar (Figura 32) y pulsar la tecla suprimir.

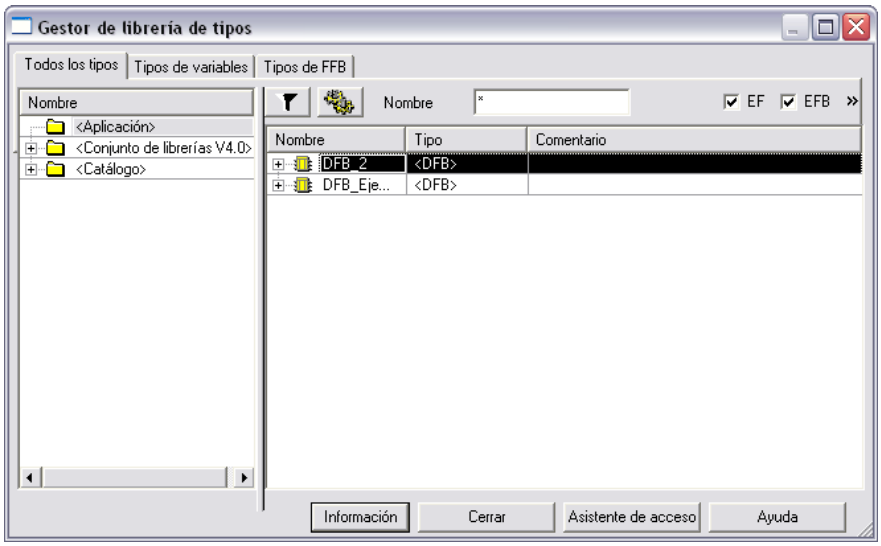


Figura 32 – Eliminación de DFB ubicado en librería

4.7 Eliminación de instancias de DFB

Para eliminar las instancias de los bloques función se debe acceder a la sección correspondiente a las instancias de bloques funcionales derivados presente dentro del explorador de proyecto. Ahí pueden encontrarse las instancias de todos los bloques función (Function Block Instance: Figura 33), ser seleccionadas y eliminadas siempre que Unity esté en modo de desconexión.

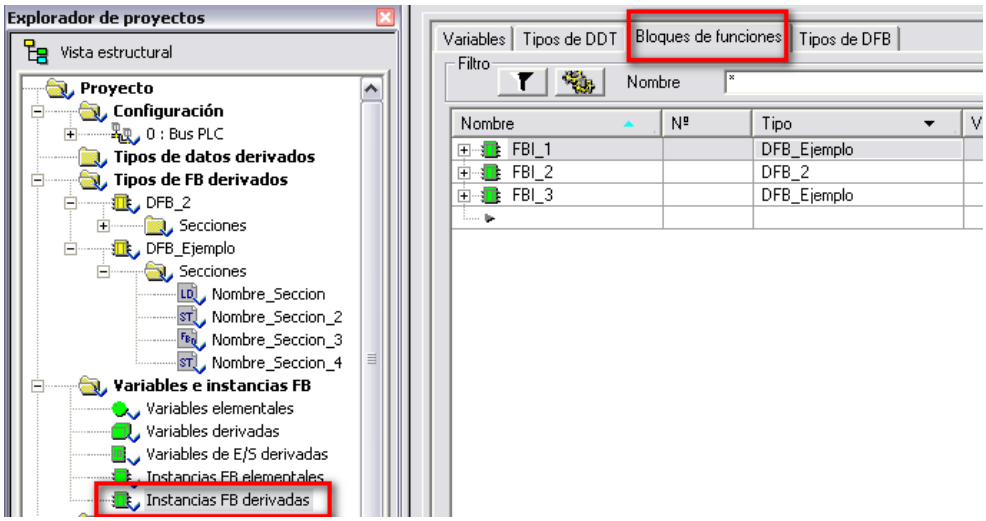


Figura 33 – Instancias de DFB

5 DEPURACIÓN Y AJUSTE

Con el paso del tiempo, tanto los sistemas electrónicos como el software han ido aumentando su complejidad, es por ello que se han ido desarrollando distintas técnicas que permitan detectar anomalías, corregir funcionalidades y optimizar el código.

El concepto de depuración, en informática, se refiere al proceso de limpieza que se realiza en un programa para identificar y corregir errores o problemas de programación. Se dice que un programa se encuentra depurado cuando no contienen errores. Mediante el proceso de depuración se realiza un seguimiento del funcionamiento del programa, observando los valores de las distintas variables así como analizando los resultados obtenidos en las operaciones.

Para simplificar la fase de depuración es conveniente la utilización de herramientas destinadas para este fin. A través de estas herramientas se podrá intervenir durante la ejecución de un programa para saber cómo se lleva a cabo la ejecución del mismo hasta ese momento.

Debido a la idiosincrasia de los dispositivos de este estudio dispondremos de herramientas específicas para depuración como es la animación de programas. Ésta será diferente según el lenguaje usado en cada sección del programa. Antes de analizar la depuración de los DFB será necesario conocer los tipos de animaciones de cada lenguaje y las medidas preventivas necesarias ante reacciones imprevisibles.

5.1 Consideraciones generales

Durante el proceso de depuración, dependiendo de las operaciones que el usuario realice debido a la posible problemática ocasionada debido a la posible problemática ocasionada, se deberán tomar en consideración ciertas precauciones. A continuación se comentan algunas de estas medidas.

5.1.1 Modificación en modalidad de ejecución

Mientras se está ejecutando un programa en el PLC, será posible realizar modificaciones en el proyecto. Estos cambios se aplicarán inmediatamente al proceso de ejecución por lo que el programador deberá asegurarse de que éste no se vea perjudicado.

5.1.2 Detención de un PLC en modalidad de depuración

Mientras se está ejecutando un programa en el PLC, será posible su detención. A efectos prácticos la detención de la ejecución se producirá al final del ciclo de la tarea maestra.

Para asegurar el correcto funcionamiento de la aplicación será necesario comprobar que tanto puntos de interrupción como la modalidad paso a paso se encuentren desactivados antes del cambio a STOP.

5.1.3 Configuración de una tarea en STOP

Dependiendo desde donde se realice la invocación de un DFB, la instancia puede estar situada dentro de la tarea maestra, rápida o auxiliar. Mientras se está ejecutando un programa en el PLC, será posible establecer una tarea en detención. Esta parada podrá efectuarse mediante la utilización de:

- Un punto de interrupción o de la modalidad paso a paso
- La pantalla de depuración del PLC

En los PLC M340 al establecerse una tarea en detención, las salidas asociadas cambiarán de forma automática a la modalidad configurada: retorno o mantenimiento.

5.1.4 Desactivación de tareas

La desactivación de tareas tiene consecuencias distintas a la detención de éstas. Al desactivarla, las entradas/salidas se comportan de forma distinta según el tipo de PLC. El comportamiento de las entradas/salidas es distinto a cuando se detiene una tarea. El usuario deberá tener en consideración que al desactivar tareas las entradas y salidas seguirán estando activas.

Esta anulación podrá efectuarse mediante la utilización de:

- Bits de sistema:

  - %S30: Activación/desactivación de la tarea maestra. Se encontrará inicialmente a 1. Cuando el usuario establezca el bit a 0 la tarea maestra se desactivará, momento en que el código de usuario no se volverá a ejecutar. El sistema comprobará este bit al final de cada ciclo de la tarea maestra.
  - %S31: Activación/desactivación de la tarea rápida. Se encontrará a 0 hasta que el usuario cree la tarea. Se desactivará al establecerse el bit a 0.
  - %S32: Activación/desactivación de la tarea auxiliar 0. Se desactivará al establecerse el bit a 0.
  - %S33: Activación/desactivación de la tarea auxiliar 1. Se desactivará al establecerse el bit a 0.
  - %S34: Activación/desactivación de la tarea auxiliar 2. Se desactivará al establecerse el bit a 0.
  - %S35: Activación/desactivación de la tarea auxiliar 3. Se desactivará al establecerse el bit a 0.

- La pantalla de depuración del PLC

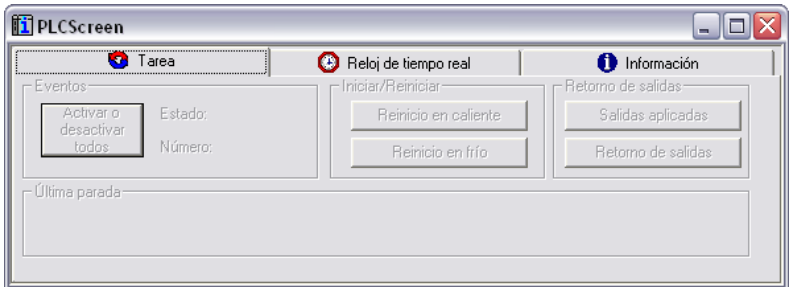


Figura 34 – Pantalla depuración del PLC no habilitada

DESACTIVACIÓN DE ENTRADAS/SALIDAS

La desactivación de entradas y/o salidas no conlleva la desactivación de la tarea.

- Desactivación de entradas: se realiza mediante la palabra de sistema %SW8. El estado inicial de este objeto del sistema es 0 y con la exclusividad de las entradas/salidas de CANopen, se puede establecer este bit en 1 o en 0.

  - %SW8.0 = 1 asignado a la tarea MAST.
  - %SW8.1 = 1 asignado a la tarea FAST.
  - %SW8.2 a 5 = 1 asignado a las tareas AUX de 0 a 3.

- Desactivación de salidas: se realiza mediante la palabra de sistema %SW9. Antes de realizarla se deberá prestar atención a la modalidad de desactivación, retorno o mantenimiento. El estado inicial de este objeto del sistema es 0 y con la exclusividad de las entradas/salidas de CANopen, se puede establecer este bit en 1 o en 0.

  - %SW9.0 = 1 asignado a la tarea MAST.
  - %SW9.1 = 1 asignado a la tarea FAST.
  - %SW9.2 a 5 = 1 asignado a las tareas AUX de 0 a 3.

5.2 Animación

La animación gráfica de las distintas secciones del programa es uno de los principales métodos que permiten al usuario (o programador) realizar, de forma más sencilla y práctica, el proceso de depuración. Debido a los lenguajes gráficos además de literales definidos en la normativa e implementados en Unity Pro, se dispondrá de la capacidad visual de comprobación de los distintos valores que adquieren las variables y parámetros.

La animación se encuentra disponible tanto desde el editor del lenguaje utilizado como mediante tablas. Su activación o desactivación vendrá dada por la activación de la opción de animación, disponible en la barra de herramientas (Figura 35) y que se encontrará activada por defecto.

Para que el equipo pueda realizar la animación mediante el editor del lenguaje será necesaria la activación del modo online y que se haya transferido el proyecto al PLC (en modalidad virtual o real). Debido al propósito de esta funcionalidad se recomienda el uso del modo de simulación.

Figura 35 – Botón animación

5.2.1 Consideraciones generales:

- La animación se gestiona sección a sección, por lo que se podrá detener o reiniciar sección a sección.

- Las variables que se actualizan muy rápidamente no se podrán visualizar en pantalla de forma correcta aunque el PLC sí que se podrá encargar de ello siempre y cuando no supere la frecuencia soportada.

- Las cadenas de más de 16 caracteres deberán ser animadas desde una tabla de animación.

- Dependiendo del lenguaje que esté siendo utilizado, se podrá visualizar el nombre de la variable a la vez que el contenido o no. Para solucionar esto existe la función información sobre herramientas que se activa al mover el cursor por las variables y muestra la siguiente información:

    - Valor de la variable
    - Tipo, nombre, dirección y comentario

5.2.2 Modo de conexión

El tipo de conexión y las funciones permitidas vendrán caracterizados por:

- La versión del proyecto abierta en el PC y la presente en el PLC
    - Modalidad integral online
    - Modalidad degradada online
- La capacidad de modificar y/o depurar el proyecto
    - Modalidad de monitorización
    - Modalidad de programación

Modo de monitorización

El usuario no podrá modificar ni depurar el proyecto si este modo se encuentra activo. Se puede acceder a la configuración de este modo desde la ruta Herramientas -> Opciones -> Conexión.

En este modo no será posible animar las funciones elementales.

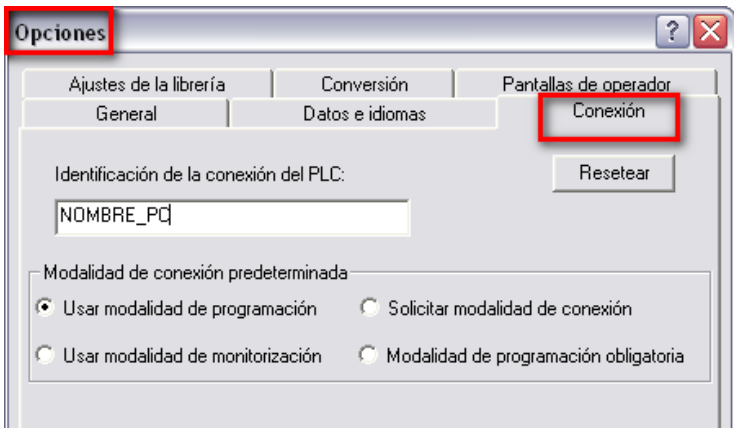


Figura 36 – Selección de modalidades

Modo de programación

El usuario podrá tanto modificar como depurar el proyecto si este modo se encuentra activo. Se puede acceder a la configuración de este modo desde la ruta Herramientas -> Opciones -> Conexión.

Modalidad integral online

El proyecto abierto es el mismo que el proyecto del PLC. Para el uso de esta modalidad será necesario asegurarse de que se encuentra activado el modo de programación.

- Depurar el proyecto. Esto se realiza mediante puntos de parada y puntos de observación.

- Modificar el proyecto. En el caso de las modificaciones se podrá generar o no el proyecto y dependiendo de esto la animación se detendrá o continuará.

    - Se genera el proyecto, las modificaciones se transfieren automáticamente y se reanuda la animación

    - No se genera el proyecto, se detiene la animación de las secciones modificadas

Modalidad degradada online

El proyecto abierto es distinto al proyecto del PLC.

5.2.3 Tipos de animación

Dependiendo del instante de la actualización de las variables de las secciones del programa, se podrán distinguir dos tipos de animación:

- Estándar: Las variables de la sección que se encuentre activa se actualizarán tras la tarea maestra.

- Sincronizada: Las variables de la sección que se encuentre activa se actualizarán a la vez que el elemento en que se haya situado el punto de observación. Este tipo de animación es de gran utilidad cuando se usa una variable en secciones diferentes del programa y se busca conocer su valor en una determinada ubicación.

5.3 Depuración según lenguaje

5.3.1 Servicios para depuración

- **Punto de parada**: Podrá existir tan sólo uno en el programa. Al insertarse un nuevo punto de parada se elimina el anterior. Su función es la de detener el programa en el punto especificado para poder conocer el valor de las variables y/o el funcionamiento del código. Se establecerá en la modalidad online y no pueden establecerse puntos de parada en una tarea de eventos.

El punto de parada puede implementarse (Figura 37) y eliminarse (Figura 38) en la modalidad offline pero su aplicación práctica mediante la ejecución paso a paso se verá limitada al cambio al modo online.

Figura 37 - Inserción de punto de parada

Figura 38 - Eliminación de punto de parada

- **Punto de observación**: Podrá existir tan sólo uno en el programa. Cuando no existe, las variables animadas se actualizan al terminar la tarea maestra. Su función es la de sincronizar la visualización de variables animadas con la ejecución de una instrucción para conocer su valor en el punto determinado.

La gran limitación de este servicio es el hecho de que si una variable se usa en varias secciones no se podrá conocer el valor de ésta en el punto determinado. Se establecerá en la modalidad online y no se pueden instertar puntos de observación en una tarea de eventos, como tampoco se puede modificar una sección en la que exista un punto de observación.

Figura 39 - Inserción de punto de observación

Figura 40 - Eliminación de punto de observación

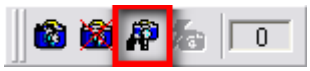


Figura 41 - Mostrar punto de observación

Figura 42 - Sincronizar tabla de animación con punto de observación

Figura 43 - Contador de paso por punto de observación

- **Ejecución paso a paso**: Consiste en la ejecución del programa elemento a elemento. Para su uso será necesaria la activación del modo online, la ejecución del programa y el establecimiento de un punto de parada.

  Cuando se requiera el retorno al punto de parada previamente establecido se reiniciará la ejecución de la tarea con el comando Continuar (Figura 44).

Figura 44 – Ejecución paso a paso: Continuar

  Mostrar paso actual localiza el paso que se esté ejecutando (Figura 45). En códigos extensos o intercalados simplifica el proceso continuo de depuración.

Figura 45 - Ejecución paso a paso: Mostrar paso actual

  Para una localización del punto del sistema en que se encuentra la depuración existe una pila de llamadas (Figura 46) tipo LIFO que almacena en orden subrutinas y DFB, incluído el intercalado de éstos.

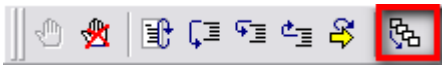


Figura 46 - Ejecución paso a paso: Pila de llamadas

- **Paso a paso por instrucciones** (Figura 47): mediante este comando se puede entrar al código e ir al próximo elemento de la sección, subrutina o DFB.

    - Si el elemento actual no contiene una llamada de subrutina o una llamada de instancia DFB, se ejecuta y se desplaza hasta el siguiente.

    - Si el elemento actual contiene una llamada de subrutina o una llamada de instancia DFB, se ejecuta y se desplaza hasta la llamada de subrutina o DFB.

    - Si el elemento actual es una llamada de subrutina o de instancia DFB, puede utilizarse para acceder al código e ir al primer elemento.

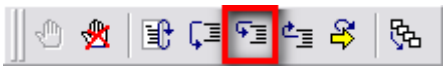


Figura 47 - Ejecución paso a paso por instrucciones

- **Paso a paso por función** (Figura 48): mediante este comando se puede entrar al código y ejecutar, en su totalidad, las funciones presentes en la sección, subrutina o DFB.

    - Si el elemento actual no es una llamada de subrutina o una llamada de instancia de DFB, se ejecuta en su totalidad y se desplaza hasta el siguiente.

    - Si el elemento actual es una llamada de subrutina o de instancia DFB, se ejecuta en su totalidad y se desplaza hasta el elemento siguiente.

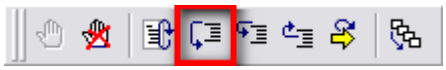


Figura 48 - Ejecución paso a paso por función

- **Paso a paso para salir** (Figura 49): mediante este comando se puede entrar al código y ejecutar en su totalidad la sección, subrutina o DFB.

    - Si el elemento actual forma parte de una instancia de subrutina o DGB, se ejecutarán todos los elementos de la subrutina o del DFB y continuará elemento a elemento después de la llamada de subrutina o DFB.

    - Si el elemento actual es una llamada de subrutina o una llamada de instancia DFB, se ejecuta todo el elemento y se desplaza hasta el siguiente.

    - Si el elemento actual es un escalón, este comando ejecuta la sección actual en su totalidad y se desplaza hasta el principio de la sección siguiente.

Figura 49 - Paso a paso para salir

5.3.2 Depuración LD

En la modalidad online y siempre y cuando el modo de animación este activo, el programa hace uso de distintos colores para ayudar al usuario a distinguir distintas situaciones de una forma más intuitiva.

Las variables booleanas a nivel alto, TRUE, podrán visualizarse en verde mientras que a nivel bajo, FALSE, se mostrarán en rojo. Las variables que representen datos númericos verán sustituidos su nombre por el valor instantáneo correspondiente y serán identificadas en amarillo.

Para que la animación de los vínculos existentes entre contactos, bobinas y bloques funcionales sea posible deberá estar activada la opción correspondiente, que en el caso de alternarse su activación y desactivación será necesario que se genere y se transifera de nuevo el proyecto al PLC.

Herramientas -> Ajustes del proyecto -> Generación de código -> Generar con animación vinculada LD

Los tipos de datos derivados, ya sean DFB o DDT que se deseen depurar deberán ser analizados en una ventana aparte de la de la sección del programa que las incluya; para ello será necesario que desde el menú contextual habilitado por el botón derecho del ratón o mediante la selección del DFB y el acceso al menú servicios se opte, en ambos casos, por la opción Detallar.

5.3.3 Depuración FBD

En la modalidad online y siempre y cuando el modo de animación este activo, el programa hace uso de distintos colores para ayudar al usuario a distinguir distintas situaciones de una forma más intuitiva.

Las variables booleanas a nivel alto, TRUE, podrán visualizarse en verde mientras que a nivel bajo, FALSE, se mostrarán en rojo. Las variables que representen datos númericos verán sustituidos su nombre por el valor instantáneo correspondiente y serán identificadas en amarillo.

Los tipos de datos derivados, ya sean DFB u DDT que se deseen depurar deberán ser analizados en una ventana aparte de la de la sección del programa que las incluya; para ello será necesario que desde el menú contextual habilitado por el botón derecho del ratón o mediante la selección del DFB y el acceso al menú servicios se opte, en ambos casos, por la opción Detallar.

5.3.4 Depuración ST/IL

En la modalidad online y siempre y cuando el modo de animación este activo el programa hace uso de distintos colores para ayudar al usuario a distinguir distintas situaciones de una forma más intuitiva.

Las variables booleanas a nivel alto, TRUE, podrán visualizarse en verde mientras que a nivel bajo, FALSE, se mostrarán en rojo. Las variables que representen datos númericos serán identificadas en amarillo.

A diferencia de otros, en los lenguajes de programación textuales no se puede identificar de forma inmediata el valor de las variables aunque sí puede conocerse al colocar el puntero sobre el elemento. Para solucionar esto es común el uso de tablas de animación.

5.4 Tablas de animación

En las tablas de animación se muestran los valores de las variables seleccionadas (siempre que el modo online esté activo o no sean entradas direccionadas) pero también se podrá modificar y forzar los valores de éstas.

El carácter temporal de la tabla de animación podrá ser elegido en la creación de la tabla, en modificaciones posteriores desde el explorador de proyectos o desde la misma tabla de animación  .

5.4.1 Creación de tablas de animación

La creación de tablas de animación podrá efectuarse desde el directorio de tablas de animación del explorador de proyectos o bien desde el mismo programa. En el primer caso, el usuario deberá elegir un nombre así como la temporalidad de la tabla (Figura 50).

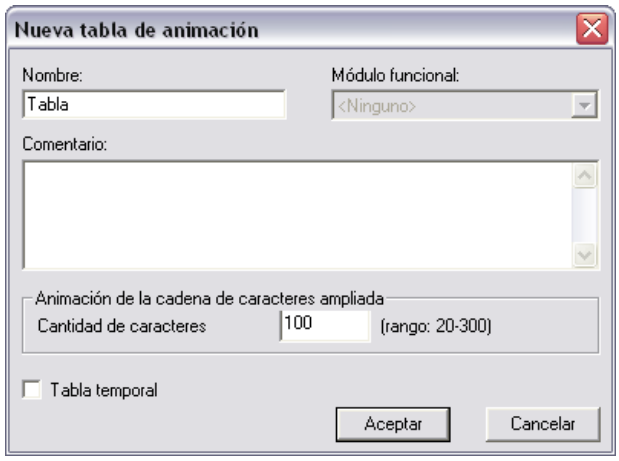


Figura 50 – Nueva tabla de animación

En el segundo caso se distinguirán varias opciones y todas ellas son accesibles desde el menú contextual creado por la pulsación del botón derecho del ratón (Figura 51). Dependiendo de si la pulsación se realiza en un espacio en blanco, en una o varias variables previamente seleccionadas o sobre un bloque funcional, en la tabla de animación creada serán incluidas automáticamente las variables asociadas a estos elementos. Las tablas de animación creadas mediante este método tienen la característica de la temporalidad, por lo que serán eliminadas al cerrar el proyecto.

La principal diferencia es que "Inicializar tabla de animación" abre una tabla de animación y si ésta no existe la crea, mientras que "Inicializar nueva tabla de animación" simplemente abre una nueva tabla.

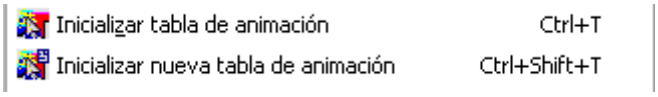


Figura 51 – Inicialización de tablas de animación

Las distintas tablas de animación creadas podrán localizarse en el directorio correspondiente a tablas de animación del explorador de proyectos (Figura 52).

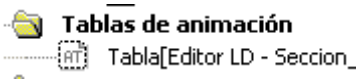


Figura 52 – Directorio de tablas de animación

5.4.2 Modificación de variables

Una vez activado este modo, será posible la modificación del valor de las variables mediante la inserción de uno nuevo en el campo valor. Para una modificación práctica más sencilla de variables booleanas se destinan los ajustes a niveles bajo y alto.

Figura 53 – Animación y modificación de variables lógicas

5.4.3 Forzado de variables

Una vez activado este modo, será posible forzar una señal booleana con direccionamiento. Se permite realizar acciones tales como el forzado a nivel bajo, alto y la cancelación del forzado 

5.4.4 Modalidad multiple 

Esta modalidad se active mediante y saca a la luz dos nuevas funcionalidades para las tablas de animación:

- La actualización de los valores en el PLC 
- El restablecimiento de los valores establecidos 

5.4.5 Menú contextual

Desde el menú contextual el usuario puede acceder a nuevas opciones como son la conversión de todas las tablas temporales en permanentes, la eliminación de todas las tablas temporales o la desvinculación de las tablas de animación de módulos funcionales.

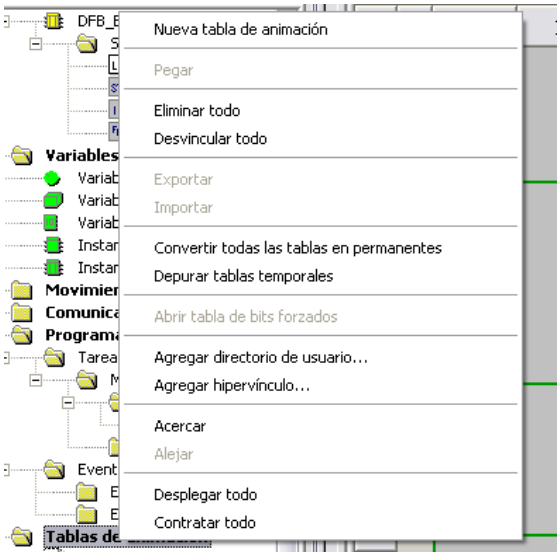


Figura 54 – Tablas de animación: menú contextual

5.4.6 Animación sincronizada

Será posible realizar la animación sincronizada de una tabla siempre y cuando se establezca un punto de observación y se seleccione la opción correspondiente: Sincronizar tabla de animación (Figura 42). Las tablas podrán animarse de acuerdo al punto de observación, identificandose esta funcionalidad mediante el símbolo de un "rayo" (Figura 55).

Figura 55 – Animación sincronizada

5.5 Pantallas de explotación

Existen cuatro tipos de objetos que pueden crearse en una pantalla gráfica[5]:

- Objetos estándar: línea, rectángulo, elipse, curva, polilínea, texto
- Imágenes: ficheros de mapa de bits con la extensión BMP o JPG
- Objetos de control (o comando): botón, casilla de verificación, campo de entrada, contador, cursor, objeto de intercambio explícito, botón de exploración de pantalla
- Objetos compuestos: grupo de objetos de tres tipos anteriores, creados por el usuario o a partir de una librería de objetos

Las pantallas de operador proporcionan un método de comunicación del usuario con el programa cuando está en funcionamiento. Los objetos disponen de diversas funcionalidades que, entre otros menesteres, le permiten al usuario realizar una depuración del programa. Por ejemplo, mediante rectángulos animados es posible realizar gráficos de tendencias, los cuales proporcionarán una representación gráfica del estado de una variable o gráficos de barras.

El acceso a las pantallas de explotación se realiza a través del directorio de pantallas de operador del explorador de proyectos. Desde el menú contextual que éste ofrece será posible crear nuevas pantallas.

Será posible la creación de familias de pantallas, las cuales se podrán asociar a módulos funcionales.

Figura 56 – Familias de pantallas de animación

[5] Fuente: [11]

5.6 Pantalla de depuración del PLC

Existe una pantalla de depuración asociada al procesador del PLC. Mediante esta será posible realizar un control de las tareas, de las modalidades de funcionamiento así como conocer información del reloj de tiempo real y del sistema. Si se interrumpiera la conexión se continuará visualizando la pantalla pero esta pasará a estar inactiva.

Activación de la pantalla

Existen dos modos de acceder a la pantalla:

- Herramientas -> Pantalla del PLC
- Haciendo doble clic sobre el procesador del PLC (Figura 58) en el editor de configuración (Figura 57) y seleccionando Animación (Figura 59).

Figura 57 – Acceso a configuración del sistema

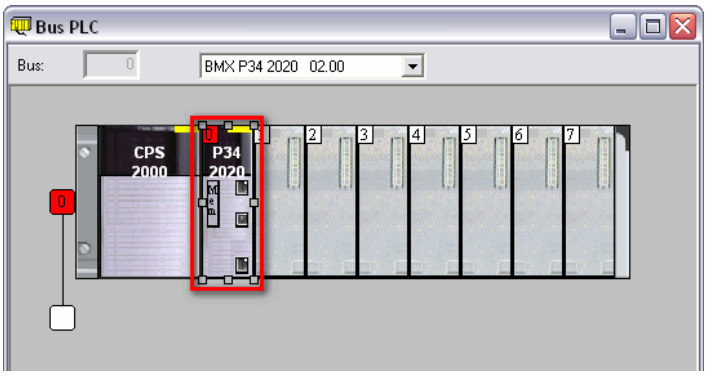


Figura 58 – Bus PLC

Figura 59 – CPU y opciones de animación

Para el uso de la pantalla es necesaria la activación de la modalidad online. Esta modalidad puede ponerse en funcionamiento mediante la activación de la conexión y el envío del proyecto al PLC o al simulador, para ello el usuario deberá seguir los siguientes pasos:

- Generar proyecto

Figura 60 – Generar y regenerar proyecto

- Conectar

Figura 61 – Conexión con PLC

- Enviar proyecto al PLC al menos una vez.

Figura 62 – Transferencia de proyecto al PLC

FUNCIONALIDADES

- **TAREA**

  - **Reinicio en caliente**: Al activarlo se establece en 1 el bit %S1. Se puede utilizar para activar un programa de inicialización parcial.

  - **Reinicio en frío**: Al activarlo se establece en 1 el bit %S0, inicializando los datos y el sistema. Se puede utilizar para activar un programa de inicialización específico.

  - **Retorno de salidas**: Al activarlo se cambian las salidas a la modalidad de retorno. Los valores de las salidas serán los valores de retorno d la modalidad STOP del PLC.

  - **Salidas aplicadas**: Se utilizan para detener la modalidad de retorno y para volver a aplicar los valores proporcionados por el programa a las salidas.

Al establecerse en STOP una tarea las salidas asociadas cambian a la modalidad configurada, ya sea retorno o mantenimiento.

Figura 63 – Pantalla de depuración del PLC: Tarea

También está disponible un menú contextual ediante el cual se pueden restablecer las modificaciones, comprobar los bits que se encuentran forzados y obtener información de impresión.

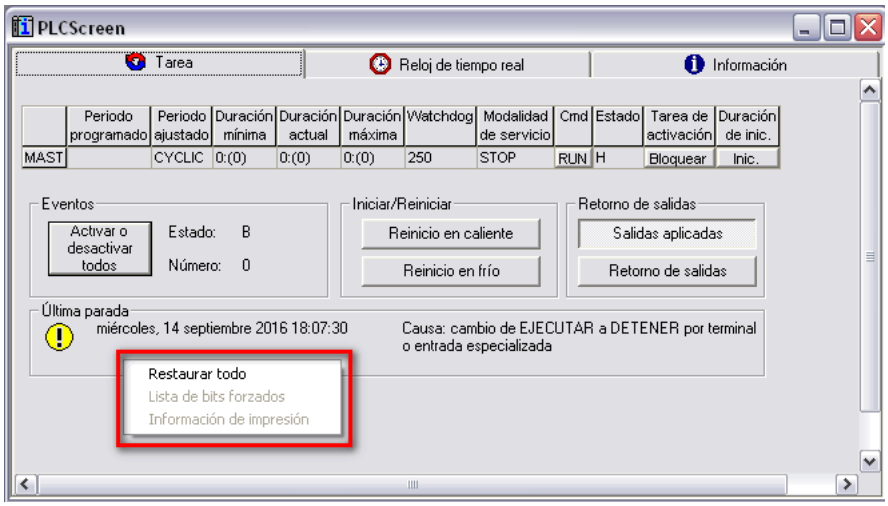


Figura 64 – Menu contextual pantalla depuración PLC

- **RELOJ DE TIEMPO REAL**

  Los procesadores de la familia BMX P34 tienen un reloj de tiempo real que controla tanto la fecha y hora actual como la de la última parada de la aplicación. Si se apaga el procesador, el reloj de tiempo real continuará funcionando durante cuatro semanas. Debido a que el procesador tramita el reloj de tiempo real para su uso es necesaria la ejecución de modalidad estándar y el PLC conectado directamente al automáta (no en modalidad de simulación).

  Esta opción permite ajustar el reloj en tiempo real según las fechas y hora ajustadas o la del PC así como visualizar la causa del error cuando no se acepte la fecha y hora.

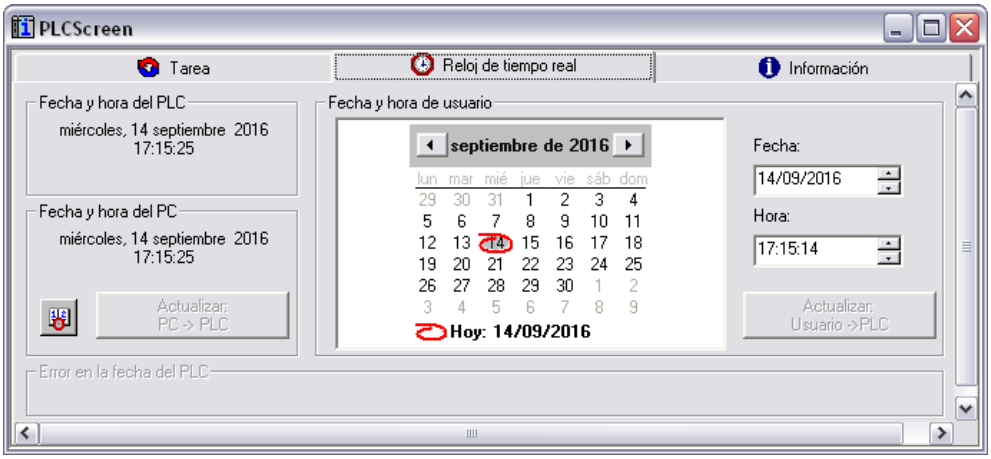


Figura 65 – Pantalla depuración del PLC: Reloj de tiempo real

- **INFORMACIÓN DEL SISTEMA**

  Descripción del sistema incorporado, éste incluye el PLC así como el programa que se incluya en él.

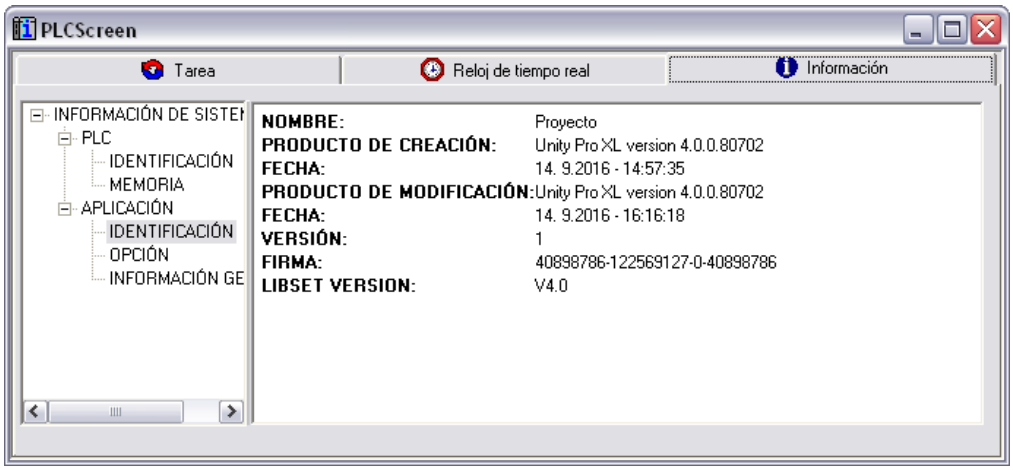


Figura 66 - Pantalla depuración del PLC: Información

Se podrá realizar una impresión accediendo a Servicios -> Información de impresión, tras haber acceder a la pantalla del PLC y seleccionar la pestaña de información.

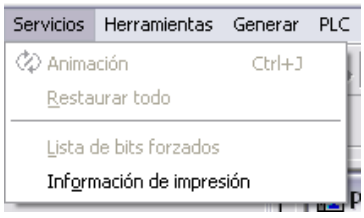


Figura 67 - Impresión de información del PLC

6 DEPURACIÓN DE DFB

La depuración mediante la animación de la vista del editor dependerá de la sintaxis del lenguaje utilizado en cada sección del bloque funcional de usuario. A continuación se detallan los distintos accesos a esta forma de depuración y las características generales.

La depuración de una instancia de DFB consiste en la animación sección a sección de las distintas instancias que componen el bloque funcional. En el caso de contar con un DFB intercalado se tendrá que detallar cada capa hasta llegar al deseado.

La animación de las instancias de los bloques funcionales derivados presentes en las distintas secciones del programa se deberá llevar a cabo mediante la opción Detallar, presente en el menú contextual y también en el menú Servicios. Con la elección de la opción "Detallar" surge un menú donde será necesaria la elección la sección específica a animar para la creación de la consiguiente ventana animada.

En las ventanas animadas mediante el detallado de bloques no se permite la modificación del código.

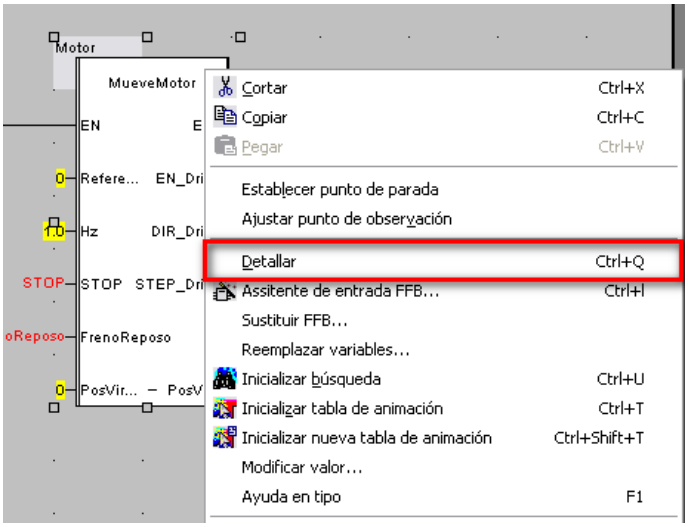


Figura 69 – Detallado de bloques función derivados

Figura 68 –Selección detallado de DFB

6.1 Tablas de animación y pantalla de operador

En el caso de querer acceder a cualquier tipo de variables de un bloque funcional desde el programa, será posible llegar a todas las capas mediante la funcionalidad del comando ".", el cual de forma similar a como si se tratatara de una estructura, permite el acceso a las variables internas.

Nombre_Instancia.Nombre_Instancia.Nombre_Variable

Esto cobra especial interés en el caso del intercalado de bloques derivados ya que permitirá entre otras cosas:

- La observación del comportamiento de las variables de cada capa del programa
- La modificación de parámetros del programa (a excpeción de las varaibles privadas)

Es singular el caso de las variables privadas. Éstas no se podrán visualizar directamente en las tablas de animación pero sí es posible su animación si se realiza el acceso a ellas mediante el comando ".".

El método más práctico para creación de tablas de animación de DFB intercalados es el que accede a los distintos niveles mediante el comando detallar (Figura 70 – Animación de DFB intercalado de nivel 1Figura 1) y la selección de inicialización de tabla en el menú contextual.

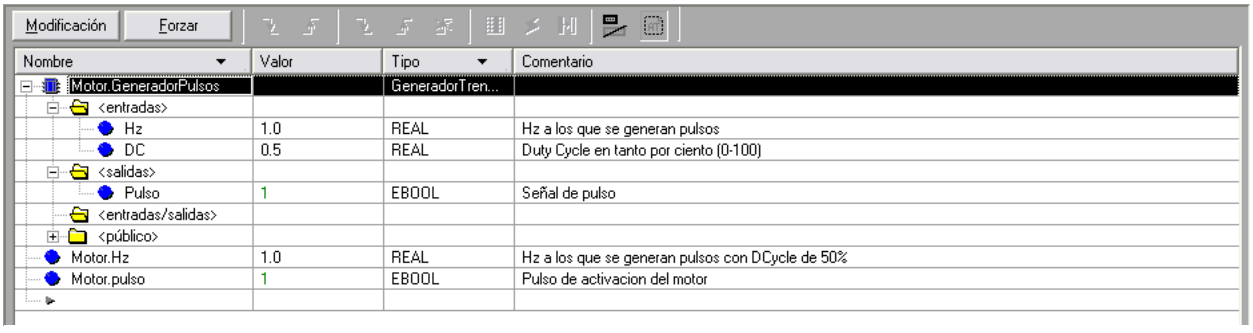


Figura 70 – Animación de DFB intercalado de nivel 1

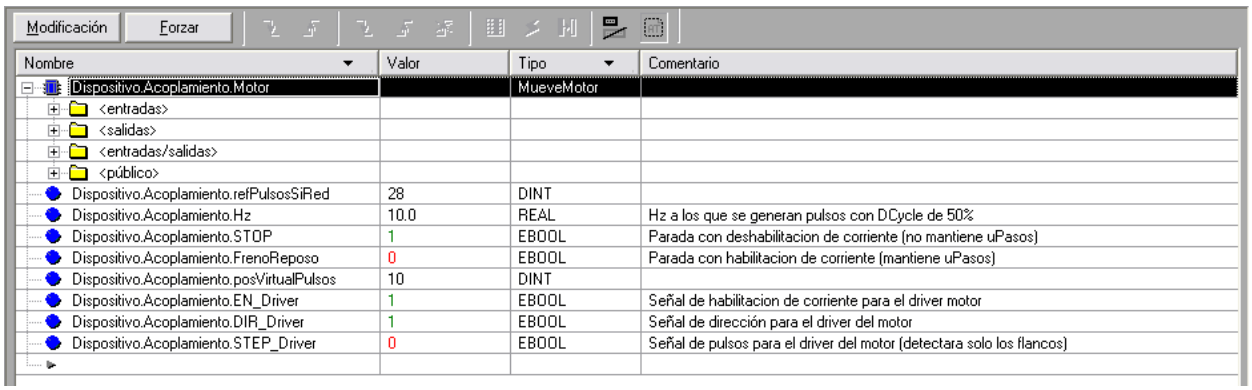


Figura 71 – Animación de DFB intercalado de nivel 2

Otro de los métodos mediante los cuales el usuario puede interactuar con el programa, además de las tablas de animación, son las pantallas de operador. Mediante estas pantallas se pueden introducir argumentos de entrada, visualizar argumentos de salida, modificar variables públicas y visualizar el valor de las variables privadas cuyo intercalado aparezca como variable privada.

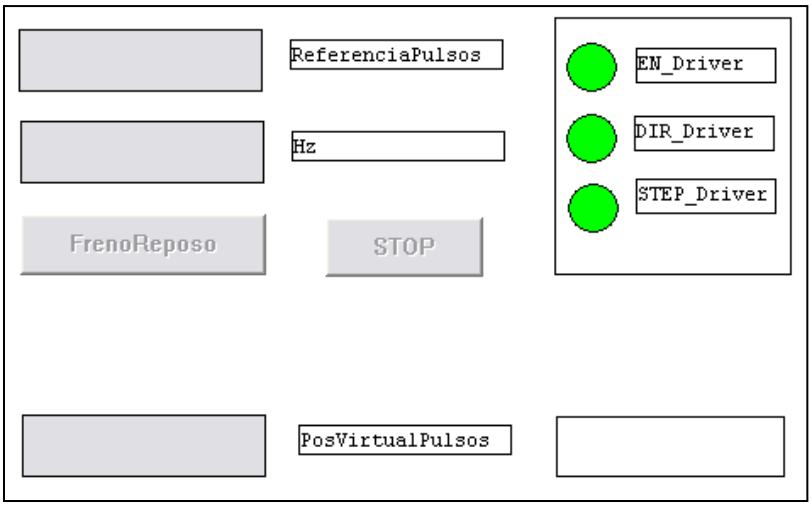


Figura 72 – Pantalla de operador

6.2 Punto de observación

Su función es la sincronización de la visualización de variables animadas con la ejecución de una instrucción para conocer su valor en el punto determinado.

Mediante el acceso a los DFB intercalados según el detallado del bloque derivado será posible la sincronización de las tablas de animación creadas con el punto de observación establecido. Tanto la tabla como el punto de observación deberán pertenecer a la misma sección, ya que las variables que no cumplen esto se sincronizan con el final de la tarea maestra.

6.3 Punto de parada

Su función es la de detener el programa en el punto especificado para poder conocer el valor de las variables y/o el funcionamiento del código. El depurado de bloques funcionales derivados no se ve afectado por ninguna particularidad.

7 CASO PRÁCTICO

Debido a que en la célula de fabricación flexible del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Sevilla existen diversos equipos que requieren de la capacidad de posicionamiento, surge la necesidad de creación de bloques funcionales que faciliten su control. El caso práctico realizado será el de un alimentador de piezas. Este sistema estará compuesto por un motor que genera un movimiento rotacional y que, junto a poleas y correas, realizará la transmisión del movimiento, el cual se basa en una librería de posicionamiento, controlada mediante un PLC.

La función principal de los instrumentos posicionadores es conocer la ubicación de un componente y asegurar, en la medida de lo posible, la repetibilidad de sus acciones. A menos que los elementos que conforman un posicionador estén debidamente emplazados, no se podrá lograr esta funcionalidad de forma veraz. Para ello, se deben tener en cuenta algunos aspectos:

- Tipo de lazo de control del posicionador: abierto o cerrado. En el caso de estudio, y debido a los elementos sensores disponibles, no se podrá realizar la implementación de un control de lazo cerrado, por el que se minimice el error existente.

- Precisión de los elementos y precisión desead. Cuanto mayor sea la precisión requerida mayor deberán ser tanto la fiabilidad como la precisión de los diferentes elementos que lo conforman.

- Zona de trabajo y límites. El rango de funcionamiento debe ser perfectamente conocido para una correcta ubicación del sistema. Los extremos constructivos del dispositivo deben estar acotados para evitar perturbaciones constructivas y prevenir posibles percandes en el medio externo.

El dispositivo real en que se implementará la librería desarrollada se ocupará de la contabilización y tramitación de órdenes sobre distintas piezas. Mediante los motores paso a paso se gestionará la posición de cada una de las dos columnas alimentadoras de piezas que constituyen cada máquina. En cada uno de los almacenes se administrarán las piezas mediante el depósito y la extracción de un determinado tipo y tamaño de ellas. Los motivos principales de la disposición contigüa son tanto la reducción de la embergadura de la planta como la, más que posible, disminución de vibraciones debido a la simetría de la misma.

Como método de programación se recurre a un sistema estructurado por capas organizadas jerárquicamente. Este método permite valorizar los niveles previos a la capa final. Se propone por el hecho de que cada bloque podrá ser reutilizado cambiando los argumentos de entrada y logrando así mejorar la depuración, la legibilidad del código y las distintas necesidades de cada capa del programa.

7.1 Descripción física de la planta

El alimentador de piezas se encuentra integrado en una célula de fabricación flexible. Esta planta está compuesta de varias zonas de trabajo interconectadas por medio de diversas cintas transportadoras, sobre las cuales se desplazarán las bandejas portando los pallets en que se transportan las piezas.

La necesidad de un sistema de alimentado de piezas para la planta surge ante la existencia, en las zonas de trabajo, de dispensadores de bandejas y de pallets pero no de un sistema automático que proporcione los elementos que éstos podrían trasladar. A día de hoy tan sólo hay incorporado uno.

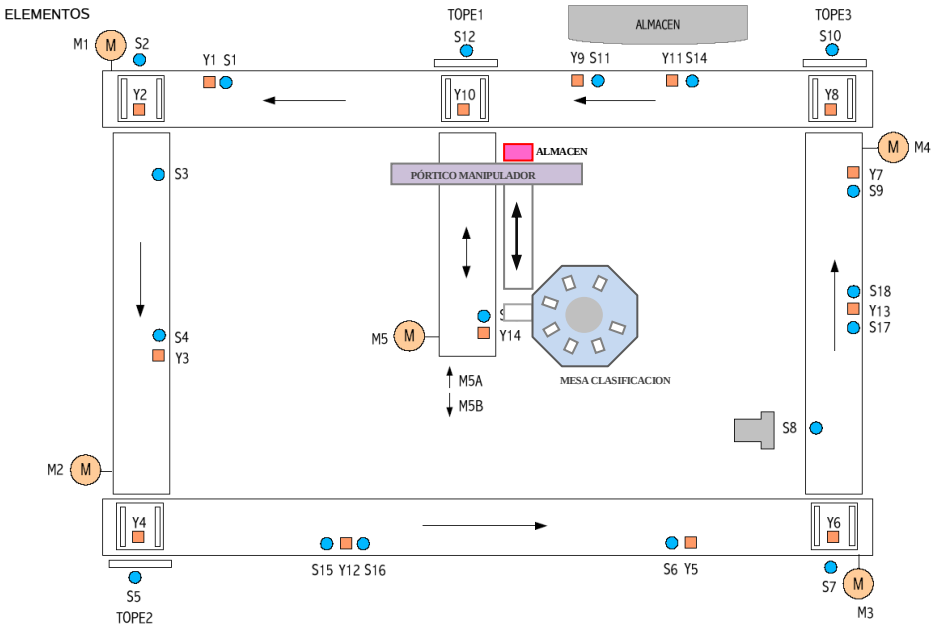


Figura 73 – Esquematico de la celúla de fabricación flexible

7.2 Descripción del sistema a automatizar

Se desarrollará un alimentador de piezas basado en un sistema de posicionamiento. Con el objetivo de la valorización de los elementos disponibles en el departamento y la adquisición de otros de bajo coste, han sido considerados necesarios los elementos de hardware abajo listados, los cuales se describirán y justificarán a posteriori.

- Motor paso a paso

  La elevación de la plataforma del gestor de piezas, se ejecutará mediante el uso y acomplamiento de un motor paso a paso. A través de correas y poleas se producirá la transformación del giro del eje en un movimiento longitudinal.

- Controlador de motores

  Debido a la potencia requerida, el movimiento del motor deberá ser gobernado por un controlador de motores. Éste, permite la posibilidad de trabajar con micro-pasos mediante selectores manuales, es decir, se podrá modificar la cantidad de pasos dados en una vuelta por el motor para conseguir mayor precisión en el posicionamiento. Debido a lo anterior, siempre y cuando no se utilicen pasos completos, el sistema deberá ser consistente con la selección de los micro-pasos a nivel de hardware y de software.

- Sensor de presencia y sensor de barrera

  La plataforma se desplazará dentro del rango de funcionamiento delimitado tanto por un sensor fotoeléctrico de barrera (ubicado en la parte superior) como por un sensor fotoeléctrico de presencia (ubicado en la parte inferior).

- Sensores de final de carrera
  Como medida de seguridad se dispondrá de sensores que asegurarán, de forma externa al bloque función, que el movimiento se realiza dentro de los límites constructivos. En la instalación realizada a día de hoy no se encuentran conectados al PLC.

Debe asegurarse la disposición de la planta, de forma que el contacto de fin de carrera no se active antes de que la plataforma llegue a provocar la activación del sensor de barrera.

Figura 74 – ALIMENTADOR DE PIEZAS

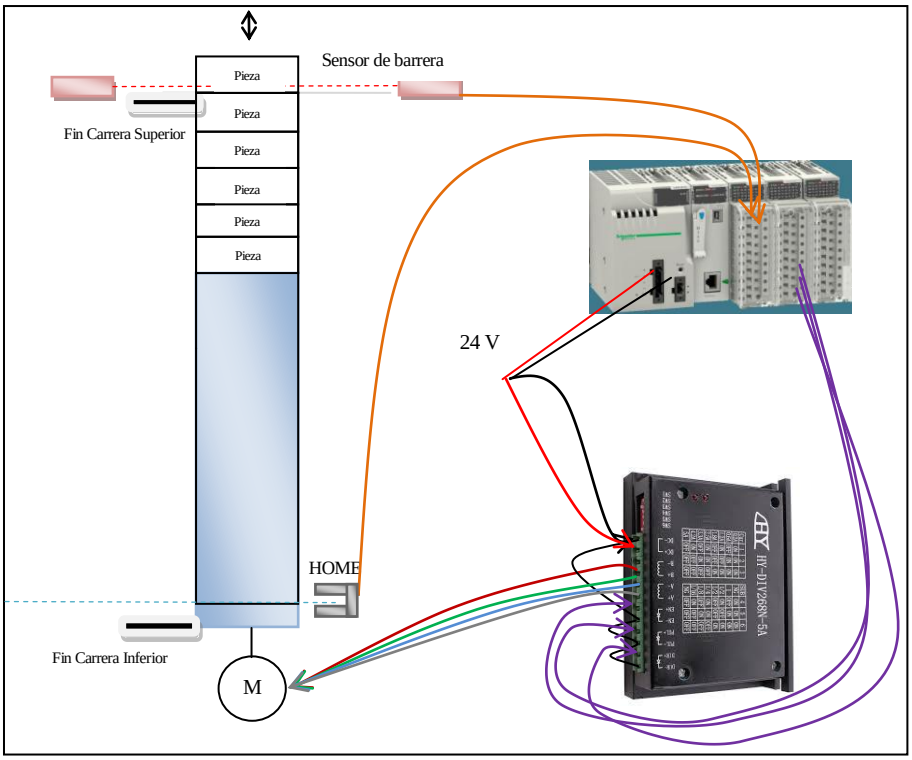


Figura 75 - Esquema del sistema

7.2.1 Motor PaP

Un motor paso a paso es aquel que gira un determinado ángulo (paso) cuando se aplican entre los extremos de sus bobinas las tensiones adecuadas. El sentido y la velocidad de giro dependen, respectivamente, de la secuencia de activación de las bobinas que forman el estator del motor y de la frecuencia de conmutación de las tensiones en los extremos de dichas bobinas.

Se define el ángulo de paso como el ángulo que describe el eje del motor (rotor) al aplicarle un impulso y puede variar de 1˚ a 90˚ según el motor; es una característica muy importante pues indica la precisión que puede llegar a tener. El ángulo recorrido mediante cada paso es 1.8º, dándose 200 pasos por vuelta.

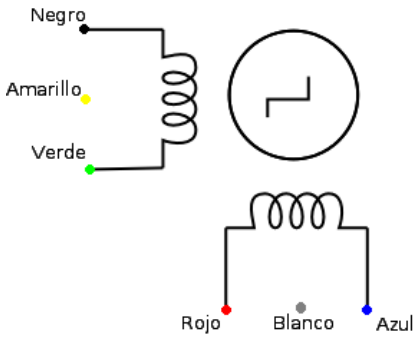


Figura 76 – Motor PaP bipolar

En general, los motores paso a paso presentan las siguientes ventajas:

- Insensibilidad a vibraciones, variaciones de tensión y temperatura
- Movimientos muy precisos
- Frecuencia de trabajo variable
- Vida útil superior a la de otros tipos de motores

Los motores híbridos se distinguen del resto en que tienen varios dientes en el estator y en el rotor, disponiendo además en el rotor de un imán concéntrico magnetizado axialmente alrededor de su eje. Esta configuración es una mezcla de los otros dos tipos de motores paso a paso, los motores de reluctancia variable y los de magnetización variable. Por lo general suelen ser más difíciles de controlar debido a que necesitan de circuitos de control y de potencia más complejos aunque esto se verá solucionado con un circuito específico de control.

El movimiento del motor marcará, gracias al sistema de polea y correa, el movimiento de la plataforma del alimentador de piezas. Para la finalidad desesada, el conexionado del motor PaP no requiere del terminal amarillo ni blanco, éstos forman a un mismo cable común y pertenecen a los extremos de cada fase. No son necesarios debido a que la aplicación a la que se destina el motor no requerirá de par elevado.

Se dispone del motor paso a paso híbrido **57BYGH420** de la empresa Wantai Motor, el cual presenta las especificaciones mostradas en Tabla 10, Tabla 9 y Figura 77 Tabla 10 - Motor: Características eléctricas:

Precisión paso	± 5%
Temperatura máxima	80 °C
Rango temperatura ambiente	-20° ~ 50°C
Resistencia de aislamiento	100 MΩ Min 500 V DC
Fuerza del dieléctrico	500 V AC 1 minuto

Tabla 9 - Motor: Características generales

Angulo de paso (°)	Voltaje (V)	Corriente (A)	Resist. (Ω/fase)	Inductancia (mH/fase)	Par retenedor (N·cm)	Par detención (N·cm)	Numero cables	Inercia rotor ($g \cdot cm^2$)	Longitud (mm)	Peso motor (kg)
1.8	3.6	2	1.8	2.5	90	3.5	6	300	56	0.7

Tabla 10 - Motor: Características eléctricas

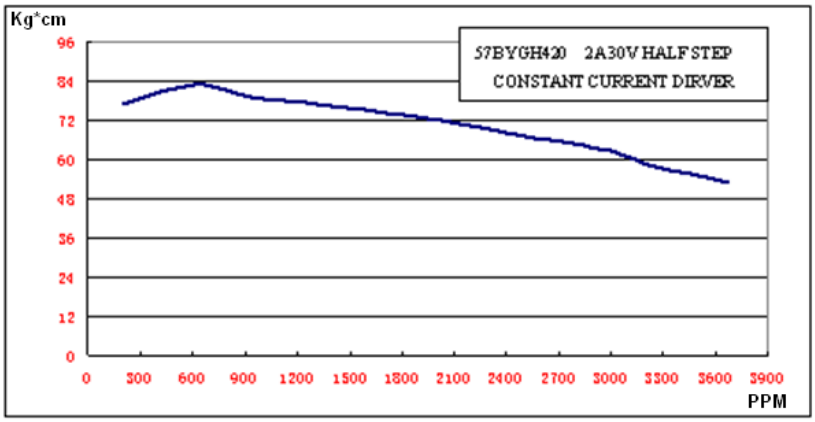


Figura 77 – Motor: Característica pulso - par

7.2.2 Driver de motores PaP

El driver **HY-DIV268N-5** es un controlador de motores paso a paso híbridos bipolares. Este dispositivo es un potente amplificador que permite al motor funcionar correctamente convirtiendo la señal de pulso (enviada de forma externa al driver) en un desplazamiento angular del motor. La velocidad del motor es proporcional a la frecuencia del pulso, consiguiendo de esta forma regular la velocidad con bastante precisión. El contaje del número de pulsos permite obtener un posicionamiento preciso.

El controlador requiere, para su correcto funcionamiento, de al menos tres señales de control. Debido al tipo de conexionado realizado, tan sólo serán necesarias tres. Las otras señales de control, **PUL-**, **DIR-**, **EN-,** se conectarán entre sí a tierra (GND) realizando una conexión de cátodo común.

- **PUL+**: señal de impulso
- **DIR+**: señal de dirección
- **EN+**: señal de habilitación

La activación (ENABLE) de los motores paso a paso se realiza mediante un contacto normalmente cerrado

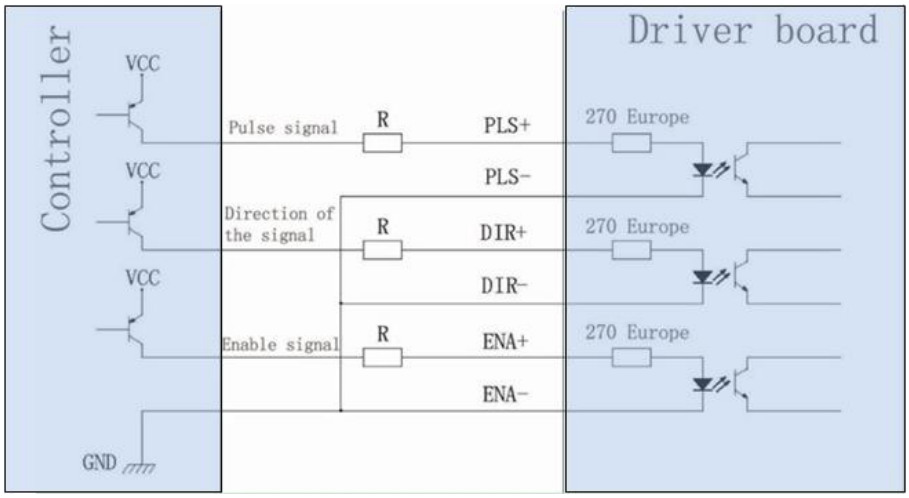


Figura 78 – Driver: Esquemático

El sistema está diseñado para su uso con alimentación a 5 V, por ello se dispone de una R pequeña. Cuando se trabaja con tensiones elevadas se debe reconsiderar la protección del diodo, para solucionarlo con se pondrán resistencias según la alimentación con la que se trabaje (teniendo en cuenta que las resistencias deben disipar más de 1/8W):

- Con Vcc = 24 V es necesario poner una resistencia R de 2kΩ
- Con Vcc = 12 V es necesario poner una resistencia R de 1kΩ

Esto es debido a la mayor protección que necesita el diodo del optoacoplador al aumentar el voltaje. El modelo del optoacoplador es el PC817.

Si consideramos como límite de cálculo el valor máximo posible de la corriente de polarización en directa obtenemos:

$$I_F \leq 50\,mA \quad , \quad V_{CC} = 24V \quad , \quad V_F = 1.4V$$
$$I_F = \frac{V}{R_i} = \frac{V_{CC} - V_F - V_{CE}}{R_i} \quad -> \quad \frac{24 - 1.4 - 0.2}{0.05} = \boxed{448\,\Omega \leq R_i}$$

Considerando un valor intermedio de corriente de polarización directa:

$$I_F \leq 20\,mA \quad , \quad V_{CC} = 24V \quad , \quad V_F = 1.2V$$
$$I_F = \frac{V}{R_i} = \frac{V_{CC} - V_F - V_{CE}}{R_i} \quad -> \quad \frac{24 - 1.2 - 0.2}{0.02} = \boxed{1130\,\Omega \leq R_i}$$

Considerando la cota inferior máxima de 5mA:

$$I_F \geq 5\,mA \quad , \quad V_{CC} = 24V \quad , \quad V_F = 1.2V$$
$$I_F = \frac{V}{R_i} = \frac{V_{CC} - V_F - V_{CE}}{R_i} \quad -> \quad \frac{24 - 1.2 - 0.2}{0.005} = \boxed{4520\,\Omega \geq R_i}$$

En el driver ya se incluyen resistencias para cada optoacoplador, debemos descontar el valor de éstas del calculado. Entiendo que debido al deseo de trabajar con un CTR decente y no demasiado elevado se indica en la hoja de características el uso de una resistencia de 2kΩ.

El driver de motores con el que se trabajará permite ajustar el uso de micro-pasos mediante selectores. Los selectores también servirán para regular la corriente que se le transmite al motor y las posibles combinaciones vienen dadas por la Tabla 11 en ella los niveles de ON/OFF se corresponden con los interruptores (switches) ABAJO/ARRIBA y los niveles lógicos correspondientes serán LOW/HIGH (debido a conexión PNP).

Micro pasos	SW1	SW2	SW3
NG	ON	ON	ON
1	OFF	ON	ON
1/2a	ON	OFF	ON
1/2b	OFF	OFF	ON
1/4	ON	ON	OFF
1/8	OFF	ON	OFF
1/16	ON	OFF	OFF
NG	OFF	OFF	OFF

Corriente salida (A)	SW4	SW5	SW6
0.2	ON	ON	ON
0.6	ON	OFF	ON
1.2	OFF	ON	ON
1.8	OFF	OFF	ON
2.5	ON	ON	OFF
3.3	ON	OFF	OFF
4.2	OFF	ON	OFF
5	OFF	OFF	OFF

Tabla 11 Driver: Configuraciones posibles

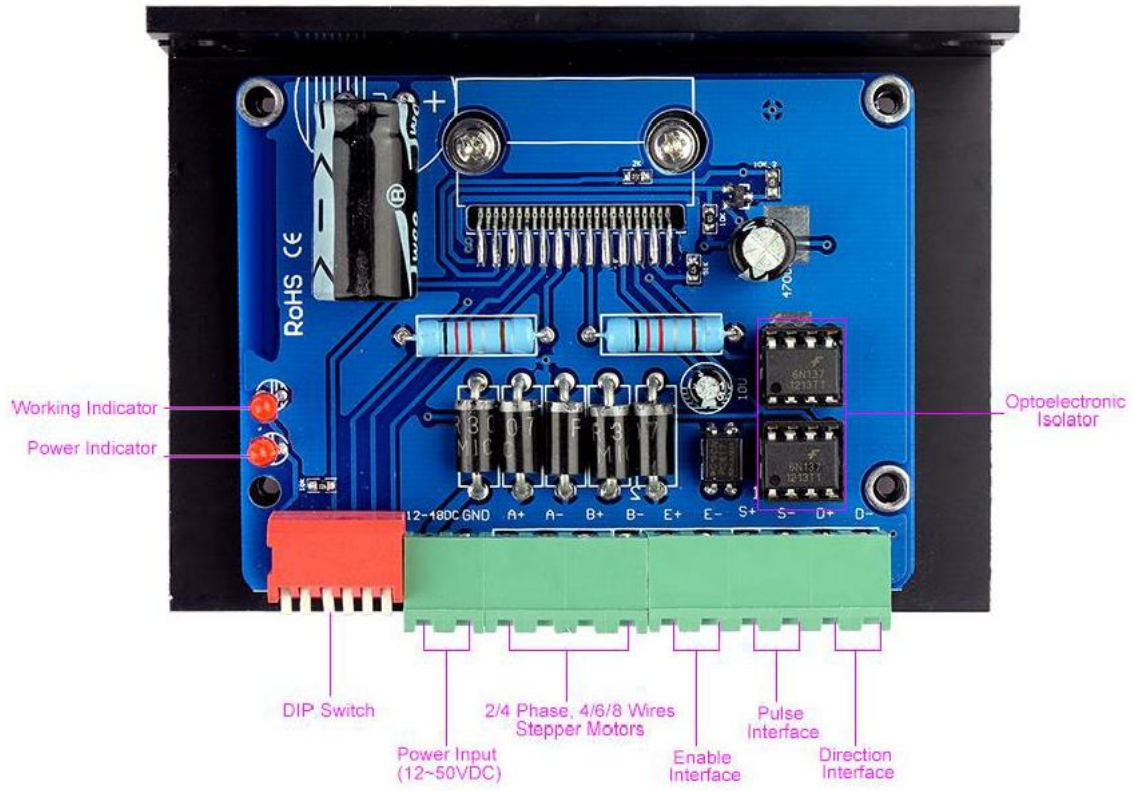


Figura 79 – Driver: Elementos

La presencia del driver de motores se justifica debido a la necesidad de disponer un elemento que permita controlar con precisión la velocidad y posición del motor. La configuración requerida deberá evitar en la medida de lo posible la pérdida de pulsos, la detección anticipada del sensor de barrera (por el movimiento de las piezas) así como una corriente compatible con la alimetación generada por la fuente y el resto de elementos del circuito.

La configuración que se usará en este proyecto implica la selección de pulsos completos y una limitación de corriente de 0.6 A.

OFF | ON | ON || OFF | ON| ON

Figura 80 – Driver: Configuración del proyecto

7.2.3 Sensor de presencia

Este sensor fotoeléctrico, modelo SFH-9500, consta de dos componentes, emisor (fotodiodo) y receptor (fototransistor). Gracias a él es posible la detección de objetos situados entre los componentes siempre y cuando no detecte el receptor el haz de luz enviado por el emiso (Figura 81).

Su inclusión en el sistema se debe a la necesidad de conocer y establecer un punto de referencia para el sistema, posteriormente nombrado como "home".

Estará ubicado en la parte inferior de la torre dispensadora, cercano al sensor de fin de carrera inferior. Deberá detectar la pieza metálica unida a la plataforma, limitando de esta forma el movimiento del asiento de las piezas.

La salida de este sensor está constituida por un contacto normalmente cerrado.

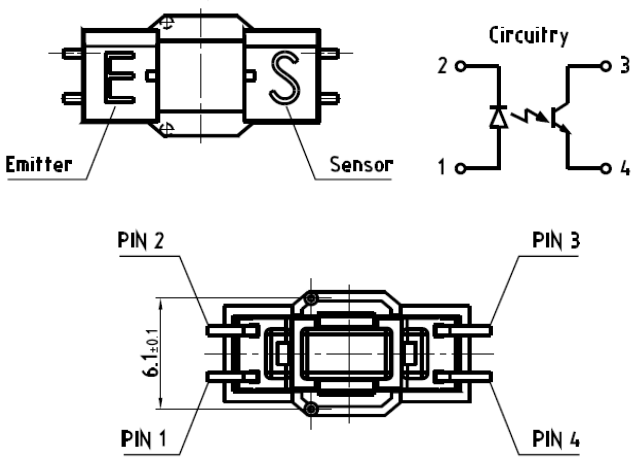


Figura 81 – Sensor de presencia: Esquemático

7.2.4 Sensor de barrera

El sensor fotoeléctrico **M18-T120P-PN** está formando por dos elementos, emisor y receptor. Funciona bajo un principio que lo hace apto para detectar cualquier tipo de objeto o persona. El diodo emisor emite pulsaciones luminosas. El receptor podrá ser un fotodiodo o un fototransistor que, en todo caso, se encargará de procesar la luz de su par, amplificando la corriente fotoeléctrica creada para comparar con un umbral de referencia y poder discernir si realmente existe o no objeto entre ambos elementos. Con él se pueden detectar objetos situados entre los componentes cuando el receptor no detecte el haz de luz enviado por el emisor.

La inclusión en el alimentador de piezas se ve justificada debido al hecho de que es necesario conocer la posición real de las piezas así como delimitar los extremos del sistema. En el sistema en que es implementado será usado para la detección de las piezas que sean depositadas o extraidas del dispensador (Figura 82).

La salida de este sensor está constituida por un contacto normalmente abierto.

Figura 82 - Sensor fotoeléctrico de barrera

7.2.5 Sensor de fin de carrera

Los interruptores de fin de carrera son elementos electromecánicos formados por un accionador vinculado a un conjunto de contactos. Cuando un elemento entra en contacto con el accionador, cerrando el circuito, este dispositivo podrá cerrar o abrir una conexión eléctrica dependiendo del modo de conexionado. De esta forma se obtendrá un contacto normalmente cerrado o abierto dependiendo del modo de conexión y el estado del accionador.

En el sistema en que puede ser implementado su función debería ser asegurar los límites de funcionamiento superior e infenior del dispositivo. Actualmente no se encuentran conectados.

Figura 83 – Sensor de fin de carrera

7.3 Especificaciones de funcionamiento

Se desea disponer de un gestor de piezas que pueda ser usado para distintas configuraciones de almacenes similares.

- Todas las piezas tendrán las mismas dimensiones.
- Mientras se ejecute una petición no se tramitarán otras.
- No será posible realizar extracciones de piezas cuando el alimentador esté vacío u operaciones de almacenado cuando esté lleno.
- El autómata debe disponer en todo momento el número de piezas almacenadas y actuar consecuentemente al alcanzar su capacidad mínima o máxima.
- Debe existir la posibilidad de realización de una puesta en marcha que reconozca el estado del sistema (número de piezas, posición de partida…). Esta puesta en marcha (home) debe de tener prioridad sobre el resto de operaciones y que se ejercerá al menos una vez: al inicio del programa.
- La plataforma podrá moverse la posición indiicada mientras que su funcionamiento no interfiera con los sensores, si se da el caso deberá actuar en consecuencia dependiendo del sensor activado.
- Los sensores de presencia y barrera marcarán la carrera de la plataforma así como el punto de extracción/almacenado de piezas.
- Se dispondrá de medidas de seguridad externas por si la plataforma rebasara su rango de trabajo.
- Se deberá considerar la fuerza ejercida por una columna de piezas. En caso de configuración de micro-pasos será necesario disponer de un sistema energizado debido a que el motor no se encontrará en posiciones estables y podrá perder pasos.
- Cuando el sistema o el usuario lo requieran, se podrá realizarse una parada de emergencia en la que el sistema dejará de estar energizado.

7.4 Bloques funcionales

La programación de este dispensador de piezas utiiza como método de programación el intercalado de bloques, debido a la reutilización y valorización de los bloques funcionales que el usuario crea. Se deberá prestar especial atención a no superar los ocho niveles de intercalado.

A continuación se describirán los distintos bloques realizados desde la capa inferior a la superior y su función dentro del programa.

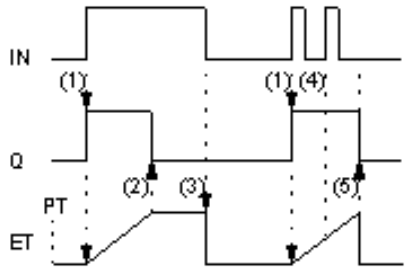
7.4.1 Generador de pulsos

Para el funcionamiento del driver es necesario un tren de pulsos continuo, que permita transmitirle al motor los pulsos con una determinada frecuencia.

El generador de pulsos puede realizarse mediante hardware o software. La generación de señales periódicas mediante software tendrá diversa problemática: estará limitada a la frecuencia que pueda soportar el automáta (60Hz sin tarjetas rápidas) y por el ciclo de scan. El uso de eventos de tipo TIMER no proporciona una solución adecuada para la generación de pulsos ya que tan sólo admite las referencias de 1,10,100 y 1000 ms. Debido a los requerimientos de baja frecuencia del sistema la generación mediante software no debería generar problemas, sin embargo siempre cabe la posibilidad de implementación de tarjetas de entradas/salidas rápidas.

Para la transmisión del tren de pulsos a una determinada frecuencia será necesaria una señal que alterne valores a nivel alto y bajo así como el porcentaje del periodo en que la señal se encontrará a nivel alto: Duty Cycle. Una de las formas más simples de conseguir esto es mediante el uso de bloques TP o Time Pulse. Esta función predefinda genera impulsos con duración definida.

Para asegurar tanto la activación como la desactivación del pulso de forma periódica se hará necesario el uso de dos bloques función.Por defecto se considerará la realización de una onda cuadrada, por lo que el DC a menos que se indique lo contrario será del 50%. En este caso, el primer bloque activará el pulso durante la mitad de la frecuencia deseada y elsegundo bloqueará la ejecución durante el resto del tiempo deseado.



(1) Si IN se convierte en "1", Q se convierte en "1" y se inicia el tiempo interno (ET).
(2) Si el tiempo interno alcanza el valor de PT, Q se convierte en "0" (independientemente de IN).
(3) El tiempo interno se detiene/restablece, si IN se convierte en "0".
(4) Si el tiempo interno aún no ha alcanzado el valor de PT, un ciclo en IN no tendrá influencia sobre el tiempo interno.
(5) Si el tiempo interno ha alcanzado el valor de PT y si IN es "0", se detiene/restablece el tiempo interno y Q se convierte en "0".

Figura 84 – Bloque función TP

7.4.2 MueveMotor

Para el movimiento del motor es necesario un generador de pulsos, implementado en este caso mediante software, así como la activación de los pulsos y el sentido de giro del motor. El encargado de transmitir estas señales al autómata es precisamente el controlador de motores.

El generador de pulsos estará en funcionamiento constante y será en esta capa donde se distingan los flancos ascendentes de éste para realizar la activación del pulso y el consiguiente movimiento del motor siempre y cuando la referencia no coincida con la posición instantánea o se haya activado la parada.

La activación del motor se producirá por medio un contacto normalmente cerrado, por lo que el sistema estará energizado por defecto. Esto provoca el planteamiento de dos posibles estados: en uno será necesario que el motor ejerza su par constructivo para bloquear el sistema al que se conecte y otro en que no sea necesario su bloqueo o se decida no recalentar/saturar un motor de bajo coste. Este estado es debido a que aunque constructivamente el motor ejerce un par mantenedor, al ser alimentado, se ubicará en sus posiciones estables (200 divisiones de 1,8º permiten recorrer el motor en una vuelta) y no la abandonará mientras las bobinas puedan impedirlo. De aquí procede la importancia de la limitación de corriente.

Deberá ser posible mantener un control de la posición del sistema, la cual se referenciará como posición virtual y sus unidades son los pasos o pulsos tramitados por el motor. El sentido del movimiento del motor estará dado por la diferencia existente entre la referencia que se desee alcanzar y la posición virtual en que se encuentre. Esta variable también podría ser interesante su valo como argumento de entrada para, por ejemplo, establecer un origen de movimientos en un momento determinado.

7.4.3 MueveAcoplamiento

Una vez que disponible un bloque que permita manejar el motor, es inmediato plantear el siguiente problema: la posición no puede manjearse en pasos realizados por el motor. El tipo de conexión/acoplamiento al que va a estar sometido el motor determinará la posición del sistema.

Normalmente se diferencian dos tipos de movimientos del efector, plano y rotatorio.Independientemente del tipo de movimiento, para conocer el desplazamiento del efector, se debe conocer el avance. Esta magnitud es el cociente entre la unidad física (distancia o grados) y los pulsos requeridos por el motor para alcanzar esa posición.

El propósito de este bloque es adaptar la señal de pasos dados por el motor a unidades físicas. La existencia de la posibilidad de aumentar la posición mediante micro-pasos sugiere la implementación de una capa intermedia entre el acoplamiento y el motor encargada del acondicionamiento. Esta capa se ocuparía de acondicionar el avance mediante una entrada que se debería actualizar manualmente (debido a que el selector del driver también es manual). Su implementación puede introducir errores debido a la falta de concordancia entre las activaciones de los switches del driver así como a la disparidad en el contaje y los posibles estados intermedios.

Con el objetivo de disponer del menor número posible de parámetros y cálculos que puedan afectar negativamente a la precisión del posicionamiento se dispondrá de una tabla experimental que permita obtener el avance (configurado para cada micro-paso) sin realizar excesivas operaciones que puedan implicar un error en la traducción de los pasos y por ende en el correcto posicionamiento.

Debido a la posibilidad de que en la traducción de la magnitud física a pulsos aparezcan números fraccionarios y que los bloques función bloqueen su funcionamiento se decide implementar un proceso de redondeo que permita minimizar el error de posicionamiento en pulsos.

Con el objetivo de facilitar la implementación de la selección de micro-pasos en el programa se adjunta la siguiente tabla en la que es posible visualizar tanto la configuración de los drivers del motor encargados de los pasos como el avance experimental necesario para la capa de acoplamiento:

Micro pasos	SW1	SW2	SW3	Avance
NG	ON	ON	ON	-
1	OFF	ON	ON	558
1/2a	ON	OFF	ON	279
1/2b	OFF	OFF	ON	279
1/4	ON	ON	OFF	140
1/8	OFF	ON	OFF	70
1/16	ON	OFF	OFF	35
NG	OFF	OFF	OFF	-

Tabla 12 – Relacion avance por micro-paso

7.4.4 MueveDispositivo

Una vez realizado el acoplamiento entre las magnitudes físicas y el movimiento del motor se deberá desarrollar la estructura física que permite el movimiento de éstos. El dispositivo estará formado por dos tipos de sensores: el sensor de presencia, el cual marcará la posición de referencia de la plataforma y los sensores de final de carrera, los cuales ayudan a asegurar el movimiento entre los límites del dispositivo.

Se desea tener un control sobre el posicionamiento de la plataforma de forma que admita referencias en posición desde el punto origen, denominado también como "HOME", por parte del usuario. Debido a esto, al iniciarse el programa el sistema debe de realizar la búsqueda del punto de referencia, además, con el fin de poder solucionar errores del sistema el usuario debe de poder solicitar la realización de la búsqueda de HOME, normalmente tras detectar algún fallo.

La búsqueda de la referencia se puede realizar de diversas maneras, se propone implementar el siguiente grafo:

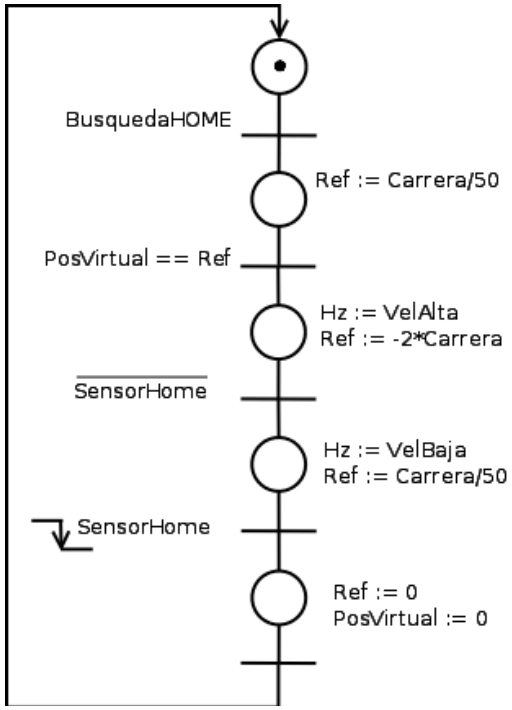


Figura 85 – Grafo para realización de HOME

Se puede apreciar como se produce un ascenso para evitar el posiconamiento entre el sensor de fin de carrera y el de presencia. Un descenso hasta que el sensor de presencia (home) detecte una interrupción. El sensor de HOME dispone de una salida normalmente cerrada, es decir, cuando se corta su valor valdrá FALSE y mientras el haz de luz no es interrumpido valdrá TRUE. Por último se produce un ascenso que permitirá el posicionamiento en el límite superior del sensor.

Para facilitar posteriores tratamientos del programa se implementarán diversas señales de estado:

- Indicador de motor en movimiento
- Indicador de plataforma fuera del rango de funcionamiento del sistema
- Indicador de búsqueda de HOME en proceso
- Indicador de fallo del sistema. Incluirá una detección de HOME falso así como la detección de los finales de carrera.

En la capa correspondiente al acoplamiento se consideró necesaria la inclusión de un método que corrigiera los posibles errores de posicionamiento surgidos por la propia naturaleza de los elementos. Para ello se propone trabajar con rangos, de esta forma los errores de posicionamiento (como puede ser la perdida de pasos) se pueden minimizar.

Por ejemplo, a la hora de conocer si el motor está o no en movimiento se sugiere la consideración de un rango de detección dado por el avance, de forma que cuando sea necesario posicionar la plataforma se deberá establecer una comparativa entre el intervalo de detección entorno a la referencia y la posición virtual.

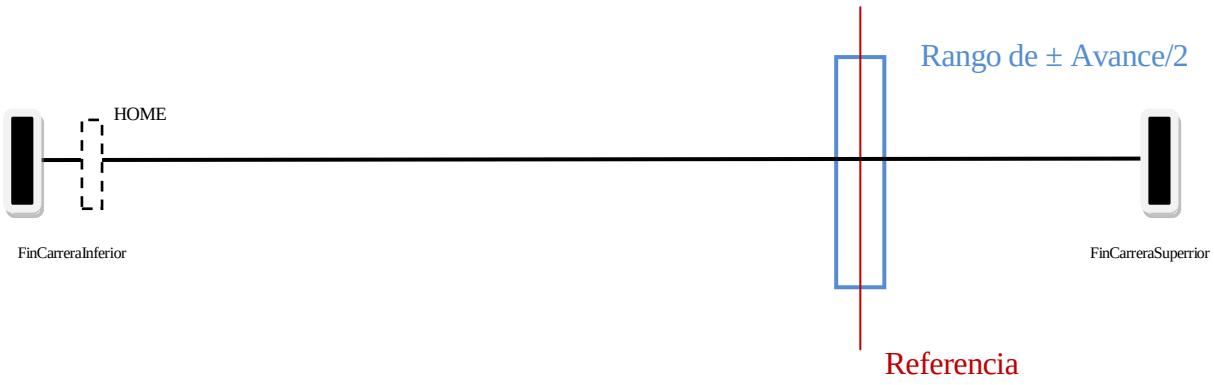


Figura 86 –Rango para detección de posición

7.4.5 AlimentadorPiezas

El objetivo del programa es diseñar un algoritmo que permita gestionar las piezas que son almacenadas y extraidas del alimentador.

Al iniciarse el programa, tanto la posición de la plataforma como el número de piezas en el sistema estará en un estado indeterminado. Será necesaria la activación de la búsquedea de referencia y el conteo de las piezas sobre la plataforma, pero debido a que el sistema puede entrar en parada, bloquearse o incluso puede desearse recalibrar el sistema cada cierto tiempo para evitar la acumulación de errores, el usuario deberá poder realizar la petición de ésta a voluntad.

Para conocer el número de piezas presente en el sistema será necesario establecer una relación entre la distancia no recorida por la plataforma y la longitud de cada pieza.

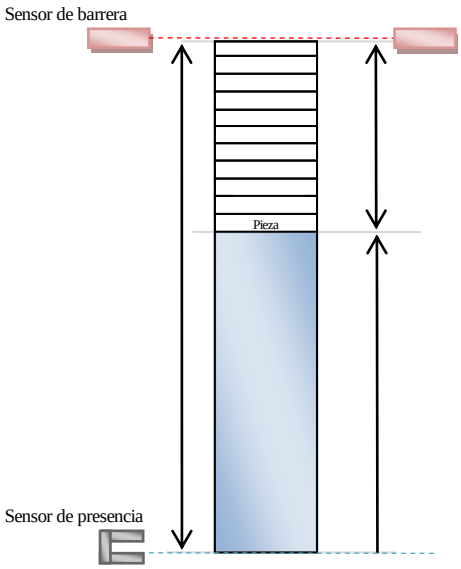


Figura 87 – Posicionamiento del alimentador de piezas

Con el objetivo de facilitar la depuración de errores y la toma de decisiones se implementará una variable de salida que contenga diversos códigos de error, en el caso del presente programa ver Tabla 13.

CODIGO	ERROR
0	No existen errores
1	Fallo de posicionamiento
2	Ocupado
3	Almacén vacío
4	Almacen lleno

Tabla 13 – Codigos de error del alimentador

8 CÓDIGO DE LAS FUNCIONES REALIZADAS

8.1 Programación GeneradorTrenPulsos

Relación de argumentos y variables del bloque

Entradas:

Nombre	Tipo	Valor	Comentario
Hz	REAL		Hz a los que se generan pulsos
DC	REAL	0.5	Duty Cycle (tanto por uno)

Salidas:

Nombre	Tipo	Valor	Comentario
Pulso	EBOOL		Pulso generado

Entradas/Salidas:

Nombre	Tipo	Valor	Comentario

Variables públicas:

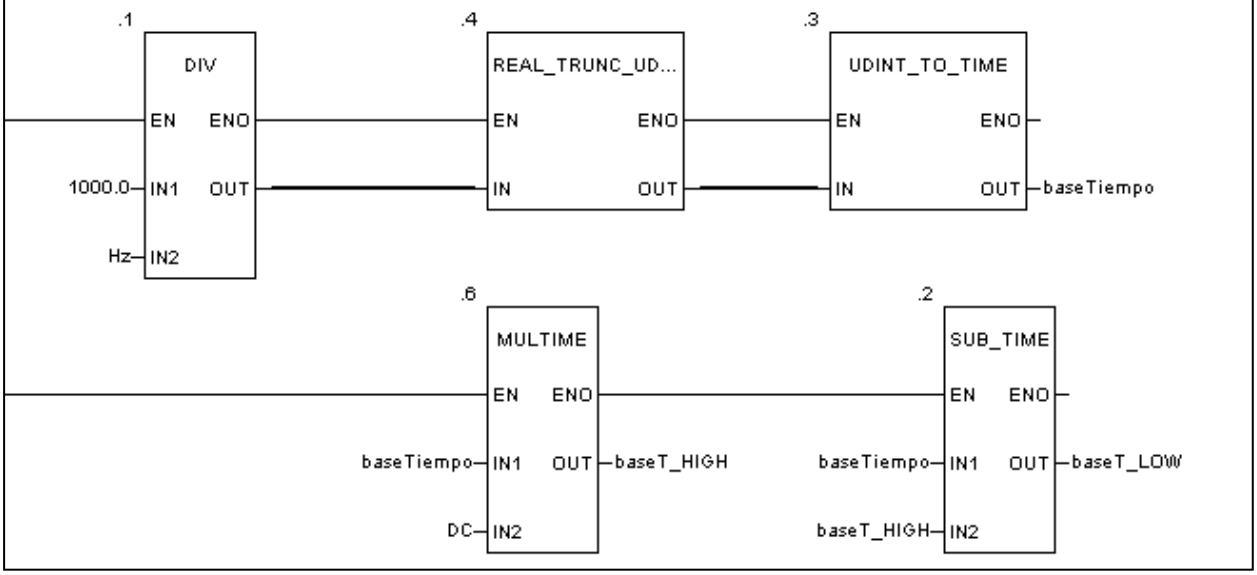
Nombre	Tipo	Valor	Comentario

Variables privadas:

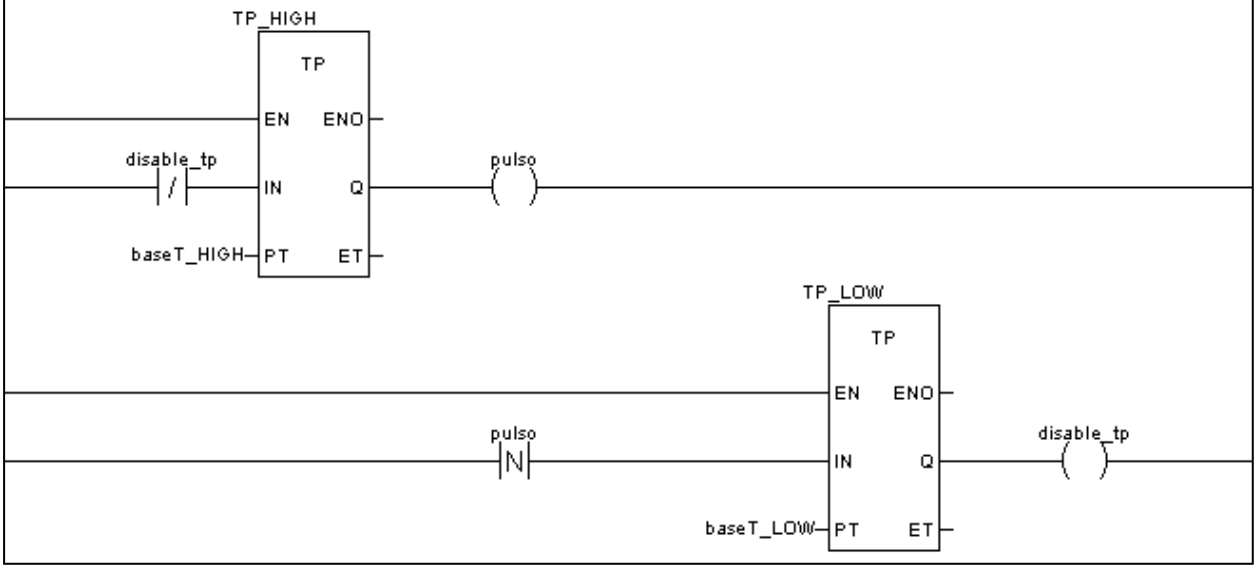
Nombre	Tipo	Valor	Comentario
TP_HIGH	TP		Bloque activador pulso
TP_LOW	TP		Bloque desactivador pulso
baseT_HIGH	TIME		Tiempo a nivel alto de la señal generada
baseT_LOW	TIME		Tiempo a nivel bajo de la señal generada
disable_tp	EBOOL		Variable bloqueadora del pulso

Seccion dedicada a operaciones de conversión de la frecuencia REAL al periodo (o base de tiempo) en ms. Se decide truncar para ir por el lado de la seguridad y no ir más rapido de lo deseado. TIPO TIME = TIPO UDINT

Conversion a variable temporal, en ms, el valor de entero doble sin signo y se consigue el tiempo que estara a nivel alto la señal: DutyCycle. Se le asigna como valor por defecto al tipo de DFB GeneradorTrenPulsos, es decir, sin ninguna entrada ni modificación de la instancia se trabaja con onda cuadrada



Sección dedicada a la generación del tren de pulsos. Se usa el bloque función TP:generación de un impulso con una duración definida. Durante parte del periodo el pulso esta nivel alto (DC) y el resto se genera una señal que desactiva el TP anterior.



8.2 Programación MueveMotor

Relación de argumentos y variables del bloque

Entradas:

Nombre	Tipo	Valor	Comentario
Hz	REAL		Hz a los que se generan pulsos
ReferenciaPulsos	DINT		Referencia de pulsos a alcanzar
FrenoReposo	EBOOL		Parada con habilitacion de corriente
STOP	EBOOL		Parada con deshabilitacion de corriente

Salidas:

Nombre	Tipo	Valor	Comentario
EN_Driver	EBOOL		Habilitacion de corriente para el driver motor
DIR_Driver	EBOOL		Dirección habilitada por el driver del motor
STEP_Driver	EBOOL		Pulsos para el driver del motor (flancos)

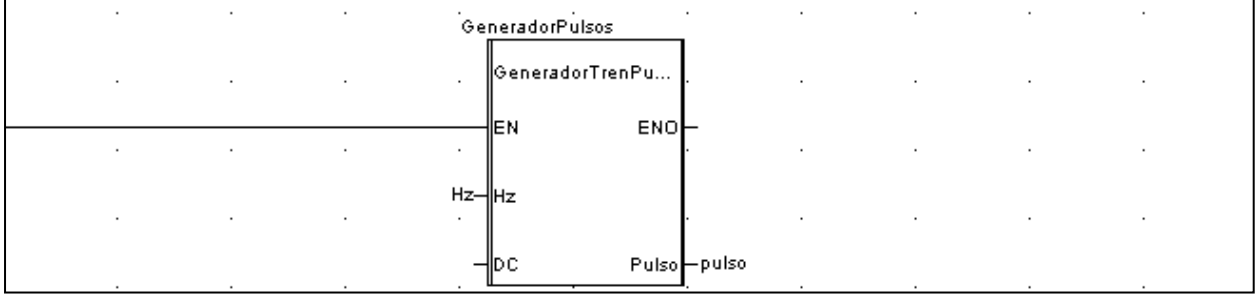
Entradas/Salidas:

Nombre	Tipo	Valor	Comentario
PosVirtualPulsos	DINT		Contaje de los pulsos dados

Variables privadas:

Nombre	Tipo	Valor	Comentario
GeneradorPulsos	GeneradorTrenPulsos		
destino	EBOOL		Flag de detección de referencia alcanzada
pulso	EBOOL		Pulso de activacion del motor

Sección dedicada a la generación de pulsos según la frecuencia introducida. Por simplicidad se considerará una señal cuadrada.



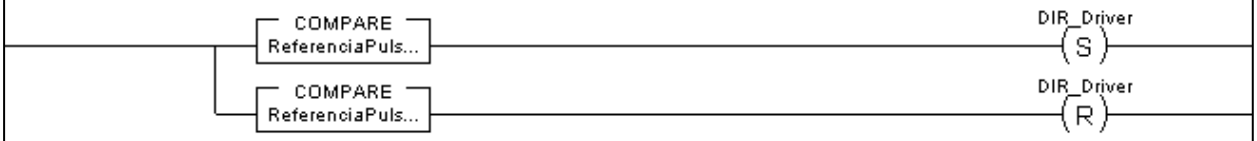
Sección dedicada al control del motor. Establecerá el sentido de giro, la comprobación del destino y la habilitación de las señales del driver de motores

Cuando ReferenciaPulsos = PosVirtualPulsos el contador ha llegado al valor deseado y por ende no seguiré aceptando pulsos del generador

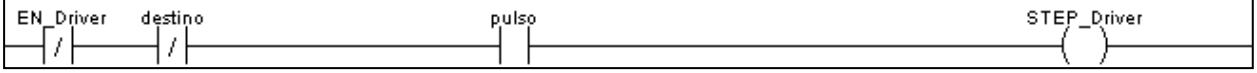
Por defecto y mientras haya alimentación el motor estará encendido. El único momento en que se deshabilitará es cuando se active STOP o cuando se llegue a destino sin estar activado el freno

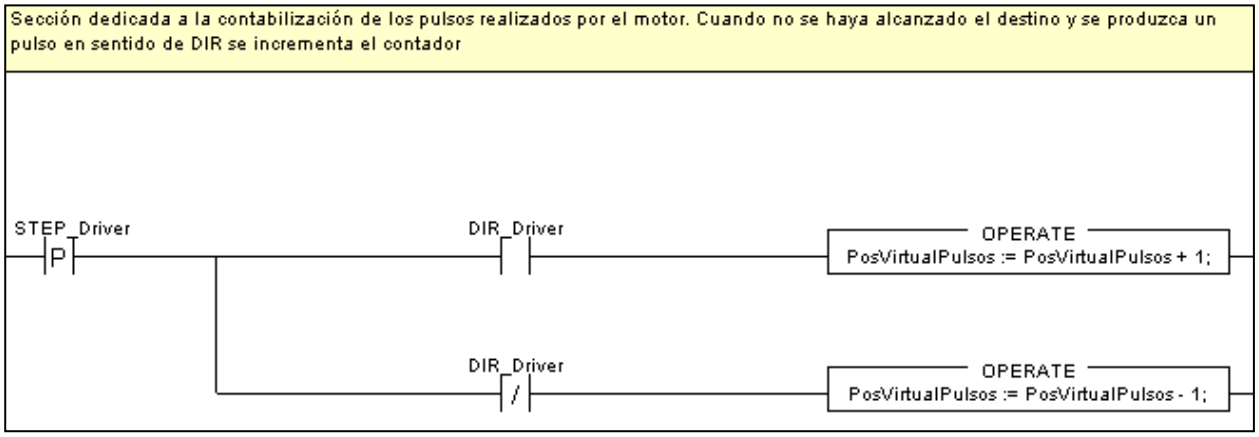
Si FrenoReposo: ENABLE = FALSE para que energize

La comparación de la Referencia con la posicióno Virtual nos permitirá identificar el sentido de giro del motor EN FUNCIONAMIENTO.



Cada vez que el tren de pulsos esté a nivel alto activaremos la salida de movimiento de pasos. El driver trabaja con flancos de subida




Sección dedicada a la contabilización de los pulsos realizados por el motor. Cuando no se haya alcanzado el destino y se produzca un pulso en sentido de DIR se incrementa el contador
STEP_Driver
P
DIR_Driver
OPERATE
PosVirtualPulsos := PosVirtualPulsos + 1;
DIR_Driver
/
OPERATE
PosVirtualPulsos := PosVirtualPulsos - 1;

8.3 Programación MueveAcoplamiento

Relación de argumentos y variables del bloque

Entradas:

Nombre	Tipo	Valor	Comentario
Hz	REAL		Hz a los que se generan pulsos
RefPosVirtual	DINT		Referencia con precision de milésimas de mm
FrenoReposo	EBOOL		Parada con habilitacion de corriente
STOP	EBOOL		Parada con deshabilitacion de corriente
Avance	UDINT		Relacion milésimas de mm/pasos.

Salidas:

Nombre	Tipo	Valor	Comentario
EN_Driver	EBOOL		Habilitacion de corriente para el driver motor
DIR_Driver	EBOOL		Dirección habilitada por el driver del motor
STEP_Driver	EBOOL		Pulsos para el driver del motor (flancos)

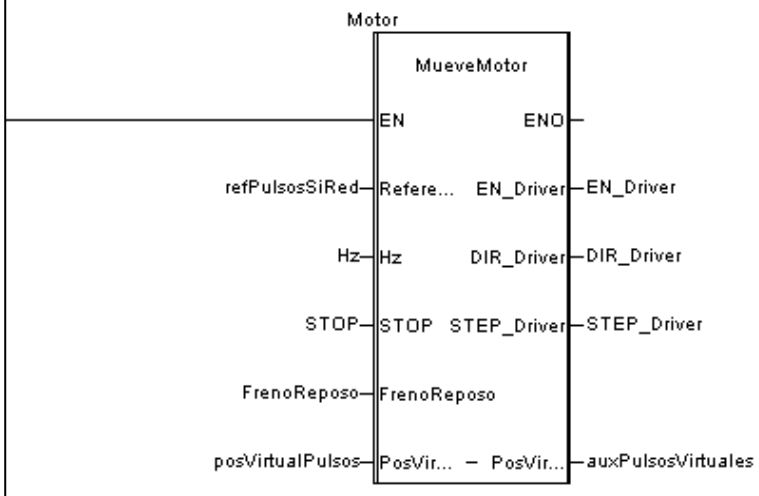
Entradas/Salidas:

Nombre	Tipo	Valor	Comentario
PosVirtual	DINT		Contaje de milésimas de mm

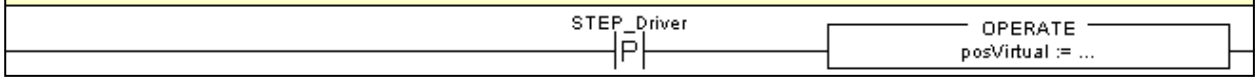
Variables privadas:

Nombre	Tipo	Valor	Comentario
Motor	MueveMotor		
PosVirtualPulsos	DINT		Contaje de los pulsos dados
auxPulsosVirtuales	DINT		Variable auxiliar para actualización de la e/s
refPulsosNoRed	DINT		Variable auxiliar de referencia no redondeada
refPulsosSiRed	DINT		Variable auxiliar de referencia si redondeada
modulo_RefPos_Avance	DINT		Resto de RefPosVirtual/Avance

Esta sección se encarga del movimiento del motor en pasos.



Debido al desacoplamiento de las funcionalidades en secciones, para poder usar PosVirtualPulsos como variable de entrada/salida y ya que se obtiene de la traduccion de otra variable de entrada/salida distinguiremos el tratamiento del argumento de entrada y del argumento de salida. En esta seccion se trata el argumento de salida, el cual se actualiza/TRADUCE cada vez que se produce una activacion de un paso.




```
(*Esta sección se encarga de la traducción de argumentos y variables*)


(*La referencia de posicion virtual es un argumento de entrada
Hay que traducir mediante el avance a pulsos del motor pero el resultado
de la operacion puede dar decimales, habra que redondear*)
refPulsosNoRed := DIV_DINT ( RefPosVirtual, UDINT_TO_DINT(Avance));



(*Realizar redondeo a través del resto. Usaremos el avance como medida cuantitativa
de la desviación máxima posible para el redondeo*)
resto_RefPos_Avance := MOD_DINT ( refPosVirtual, UDINT_TO_DINT(Avance) );


if( resto_RefPos_Avance >= (UDINT_TO_DINT(Avance)/2) ) then
refPulsosSiRed := refPulsosNoRed + 1;

        else    refPulsosSiRed := refPulsosNoRed;

end_if;



(*La posicion virtual es un argumento de entrada/salida
Como argumento de entrada habrá que traducirlo a pulsos virtuales
Los pulsos no son fraccionarios por lo que no sera necesario el redondeo*)
posVirtualPulsos:= DIV_DINT ( PosVirtual, UDINT_TO_DINT(Avance));
```

8.4 Programación MueveDispositivo

Relación de argumentos y variables del bloque

Entradas:

Nombre	Tipo	Valor	Comentario
RefPosVirtual	DINT		Referencia con precision de milésimas de mm
FrenoReposo	EBOOL		Parada con habilitacion de corriente
STOP	EBOOL		Parada con deshabilitacion de corriente
Avance	UDINT		Relacion milésimas de mm/pasos.
VelAltaHz	REAL		Velocidad alta en Hz
VelBajaHz	REAL		Velocidad baja en Hz
SensorHome	EBOOL		Sensor presencia optoacoplador
SensorFC1	EBOOL		Fin Carrera Inferior
SensorFC2	EBOOL		Fin Carrera Superior
Carrera	DINT		Recorrido de la plataforma: milésimas de mm

Salidas:

Nombre	Tipo	Valor	Comentario
EN_Driver	EBOOL		Habilitacion de corriente para el driver motor
DIR_Driver	EBOOL		Dirección habilitada por el driver del motor
STEP_Driver	EBOOL		Pulsos para el driver del motor (flancos)
FALLO	EBOOL		Indica Falso HOME y activaciones de FinCarrera
EnMov	EBOOL		La plataforma se esta moviendo
FueraRango	EBOOL		La plataforma esta fuera de la carrera normal

Entradas/Salidas:

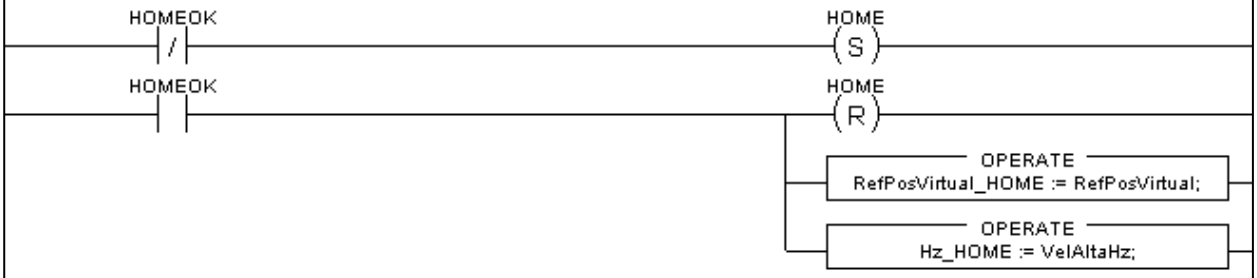
Nombre	Tipo	Valor	Comentario
PosVirtual	DINT		Contaje de milésimas de mm
HOMEOK	EBOOL		Flag indicador HOME encontrado

Variables privadas:

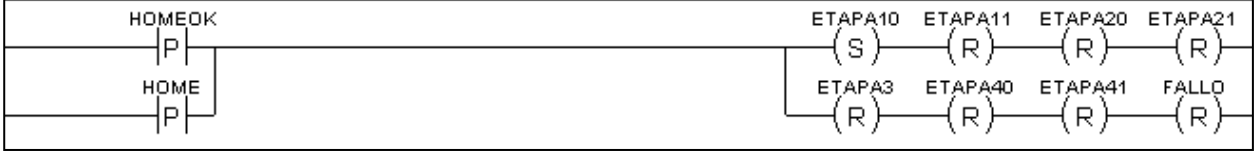
Nombre	Tipo	Valor	Comentario
Acoplamiento	MueveAcoplamiento		
RefPosVirtual_HOME	DINT		Var. Auxliar cambio referencia posicion
HOME	EBOOL		Flag indicador de realizacion de HOME
Hz_HOME	REAL		Var. auxiliar cambio referencia velocidad
ETAPA10	EBOOL	True	
ETAPA11	EBOOL		
ETAPA20	EBOOL		
ETAPA21	EBOOL		
ETAPA22	EBOOL		
ETAPA3	EBOOL		
ETAPA40	EBOOL		
ETAPA41	EBOOL		

Esta sección se encarga del control del calibrado de la posición de referencia inicial. No puedo cambiar el valor de una variable de entrada, por elloal ser el HOME una accion excepcional debere realizar un metodo excepcional mediante una variable auxiliar: RefPosVirtual_HOME

Cuando HOMEOK = false, es decir cuando el home no esta localizado, tendre que moverme para encontrarlo, por lo que se deberá seguir la sección HOME. Cuando HOMEOK = true, es decir cuando el home esta localizado, me moveré lo deseado por el usuario

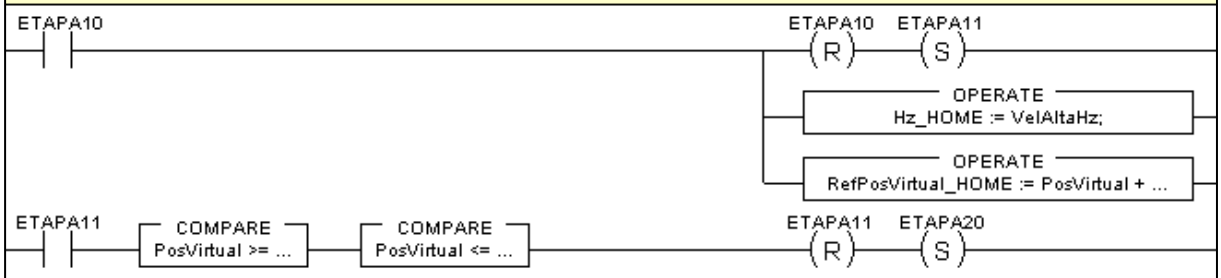


En el caso de que se produzca un pulso de subida en HOMEOK, es decir, una vez que se acabe de realizar el HOME o cuando externamente se fuerze el HOME a OK estableceremos un estado inicial que actualizará las etapas de HOME. Cuando se produzca un flanco de subida en HOME sera porque se ha producido uno de bajada en HOMEOK, es decir, hay que realizar un HOME y para ello se deberán considerar las condiciones iniciales aquí establecidas

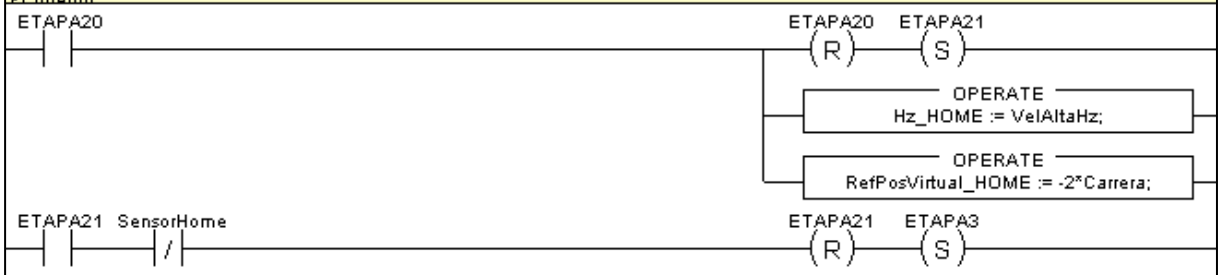


Esta sección se encarga de la traducción del grafo de control de las etapas de la busqueda de home	CONDICIONADA A HOME	FC1||HOME|| FC2	HOME -> 0 cuando se CORTA FC2,FC1 -> 1 cuando se CORTA

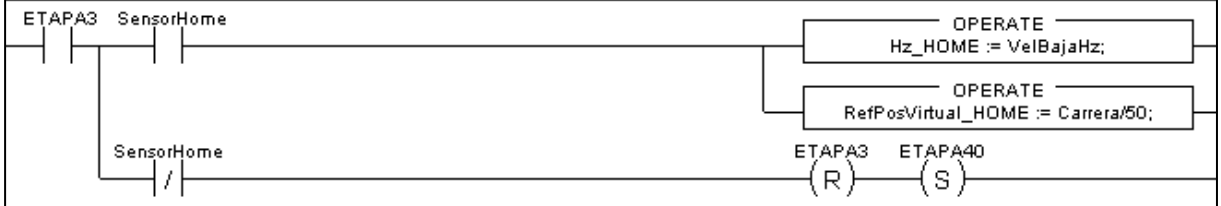
ETAPA 1.La etapa inicial consiste en asegurarnos de que no estamos entre el FC1 y el sensor HOME, para ello deberemos subir una pequeña cantidad.



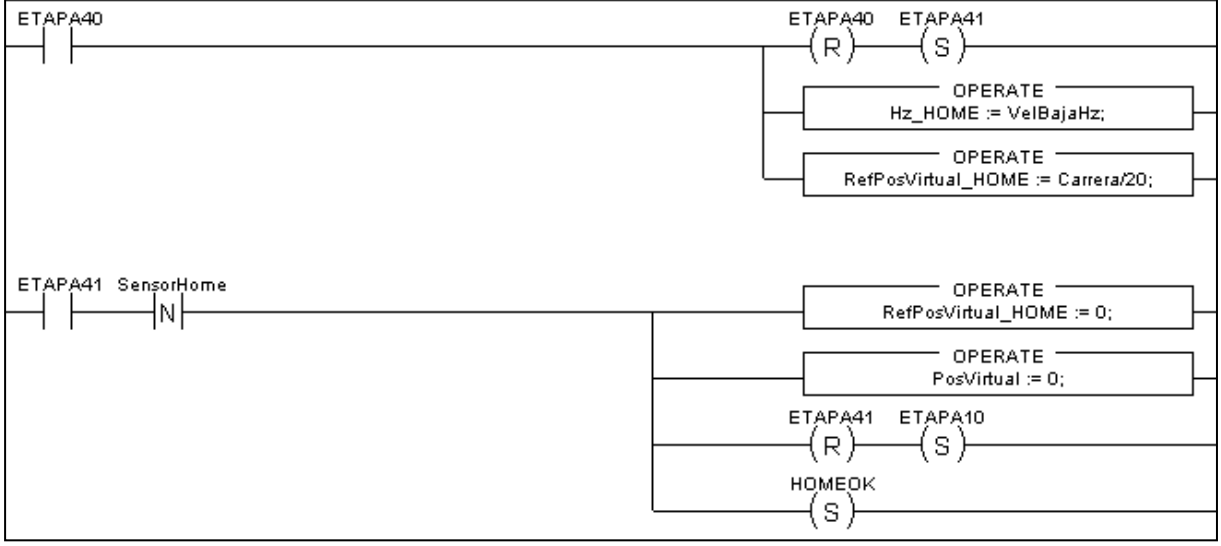
ETAPA 2. Tras subir una cierta cantidad la siguiente etapa consiste en bajar hasta encontrar el sensor de home. Si llegara a activarse el sensor de FC INFERIOR sería porque la distancia a subir no era lo suficientemente grande comparada con la separacion de sensorHome y FCInferior



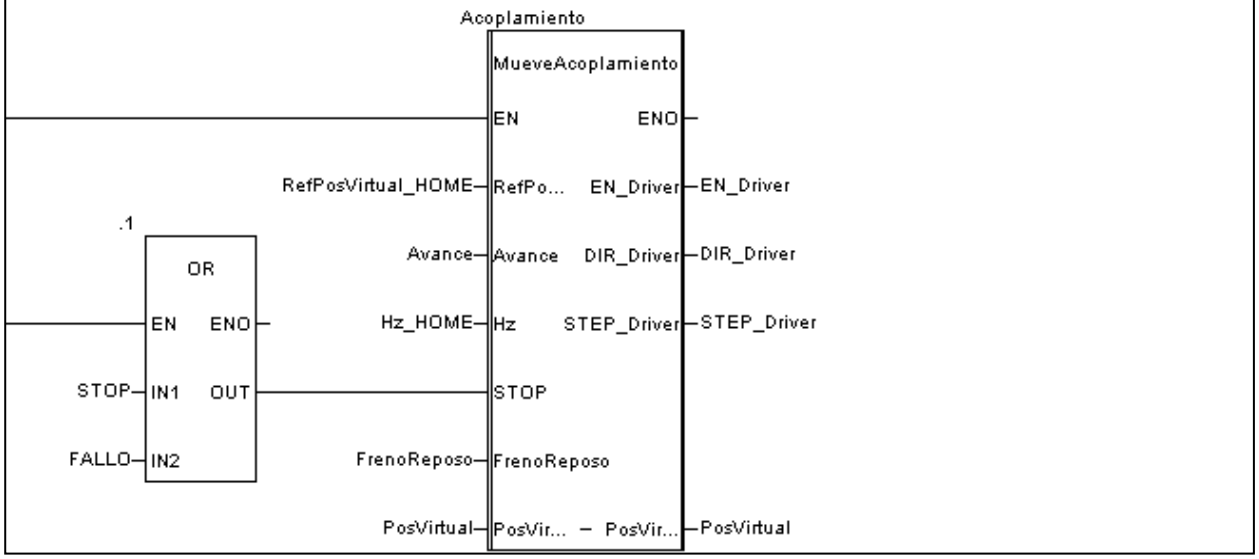
ETAPA 3. Tras haber detectado el sensor de HOME consideraré la opción de que nos hayamos pasado el sensor de HOME, por lo que ascenderemos lentamente hasta que se detecte



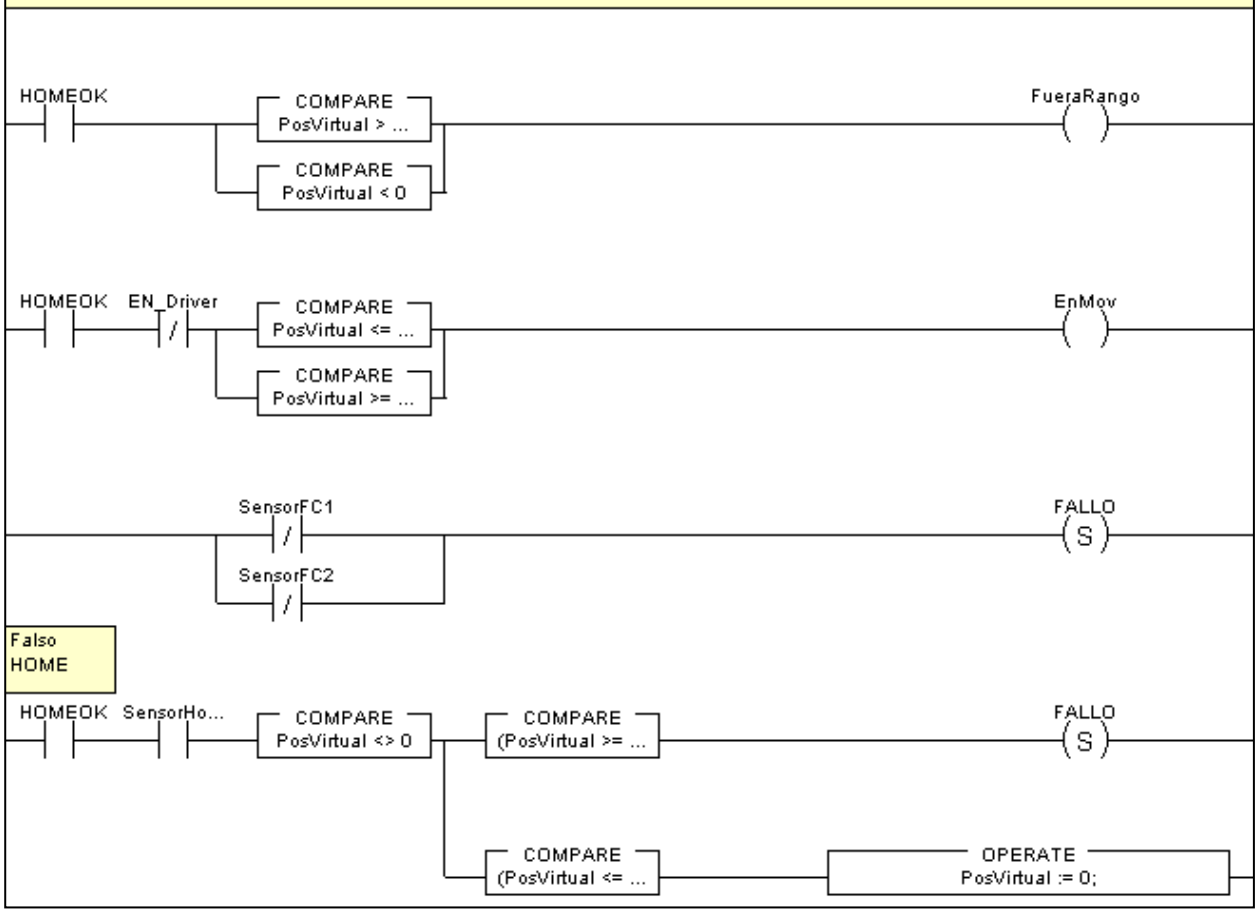
ETAPA 4. Cuando tras activarse SensorHome se detecta un flanco de bajada asumiremos que el sensor esta perfectamente LOCALIZADO. Cuando esto ocurra el HOME se habra realizado y estara OK. Hay que resetear el contador de pulsos.



Esta sección es la encargada de realizar el movimiento del motor a la referencia dedistancia indicada. La detección de un Fallo provocará la detención de energencia



Esta sección es la encargada de detectar errores de posicionamiento. En el caso de que el HOME se haya realizado correctamente (HOMEOK = true) y que se activen los sensores de FC esto indica que hay FALLO porque llegamos a los limites dl dispositivo. Por otra parte, cuando la posicionVirtual es mayor que la carrera o menor que 0 (retrasada respecto a HOME) entendemos que estamos fuera de rango y por ende tambien hay fallo. Cuando la referencia de posicion sea distinta a la posicion virtual significara que el motor esta en movimiento.



8.5 Programación AlimentadorPiezas

Relación de argumentos y variables del bloque

Entradas:

Nombre	Tipo	Valor	Comentario
STOP	EBOOL		Parada con deshabilitacion de corriente
Avance	UDINT		Relacion milésimas de mm/pasos.
Carrera	DINT		Recorrido de la plataforma: En milésimas mm
AlturaPieza	INT		Altura en milésimas de mm
VelAltaHz	REAL		Velocidad alta en Hz
VelBajaHz	REAL		Velocidad baja en Hz
SensorHome	EBOOL		Sensor presencia optoacoplador
SensorFC1	EBOOL		Fin Carrera Inferior
SensorFC2	EBOOL		Fin Carrera Superior
SensorPieza	EBOOL		Sensor presencia
ExtraePieza	EBOOL		Solicitud de piezas
AlmacenaPieza	EBOOL		Aceptacion de piezas
AutoTest	EBOOL		Establece HOME y cuenta el numero de piezas

Salidas:

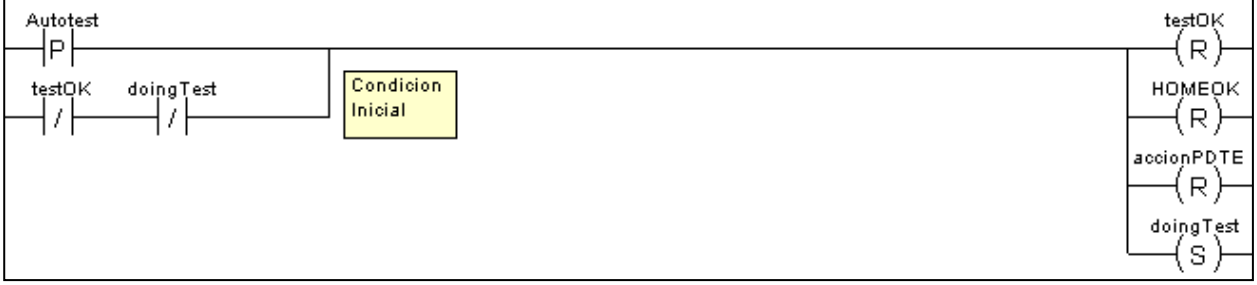
Nombre	Tipo	Valor	Comentario
EN_Driver	EBOOL		Habilitacion de corriente para el driver motor
DIR_Driver	EBOOL		Dirección habilitada por el driver del motor
STEP_Driver	EBOOL		Pulsos para el driver del motor (flancos)
NumPiezas	INT		Numero piezas del dispensador
CodigoError	INT		Codigos de error

Variables privadas:

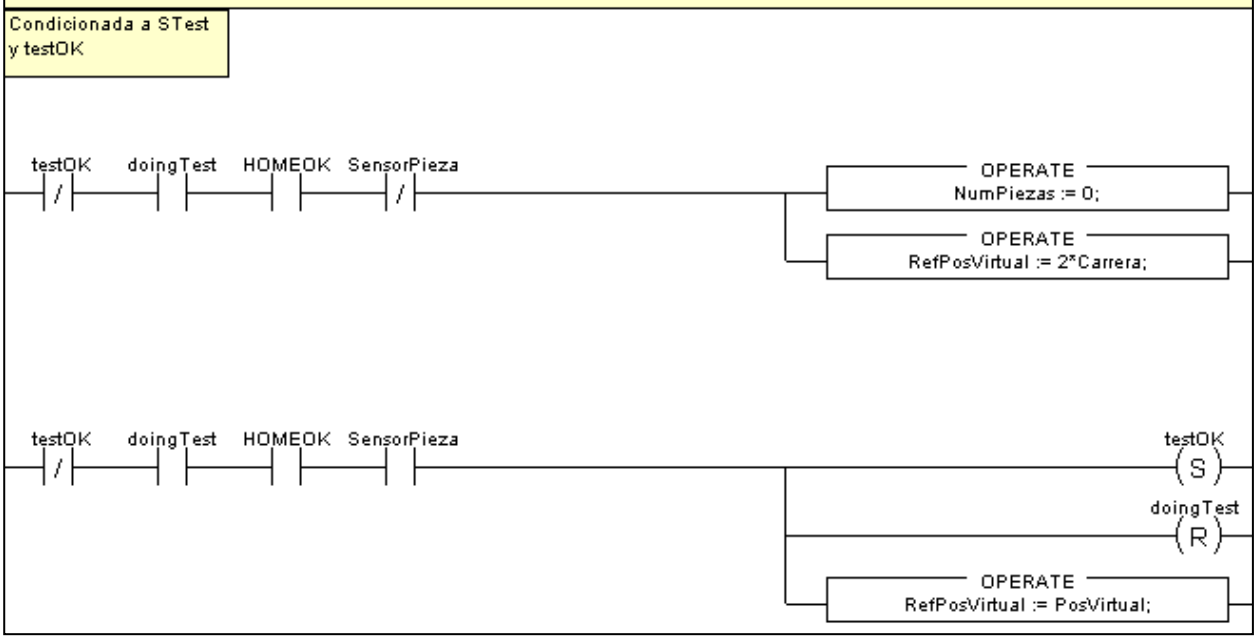
Nombre	Tipo	Valor	Comentario
Dispositivo	MueveDispositivo		
PosVirtual	DINT		Contaje de posicion en milésimas de mm
RefPosVirtual	DINT		Referencia con precision de milésimas de mm
FrenoReposo	EBOOL		Parada con habilitacion de corriente
doingTest	EBOOL		Haciendo Autotest
testOK	EBOOL		Autotest realizado correctamente
HOMEOK	EBOOL		Flag indicador HOME encontrado
accionPDTE	EBOOL		Tramitando peticion
restoPiezas	DINT		Resto para redondear el nº de piezas
auxPiezas	DINT		Division truncada del nº de piezas
restoMax	DINT		Resto para redondear la capacidad
auxMax	DINT		Division truncada de la capacidad
piezasMax	INT		Capacidad del alimentador
FueraRango	EBOOL		La plataforma esta fuera de la carrera normal
EnMov	EBOOL		La plataforma se esta moviendo
FALLO	EBOOL		Falso HOME y activaciones FinCarrera

Esta sección se encarga de controlar la realización del test al inicio del programa así como cuando se utiliza el argumento de entreada.Al comenzar el programa se deberá de realizar un AutoTest automaticamente, para que esto sea posible habilito el flag testOK. El AutoTest estará codificado en otra sección por lo que habilitamos un flag que habilite la correspondiente sección a la vez que deshabilita el HOME hasta ahora conocido. AUTOTEST := HOMEOK + CONTEOPIEZAS. Mientras se realiza el AutoTest deberemos esperar en primer lugar a que se realice correctamente la busqueda de HOME.

Deberemos asegurarnos que si se interrumpe, por cualquier causa, la extracción o el almacenado de piezas se olvida lo que se estaba haciendo.



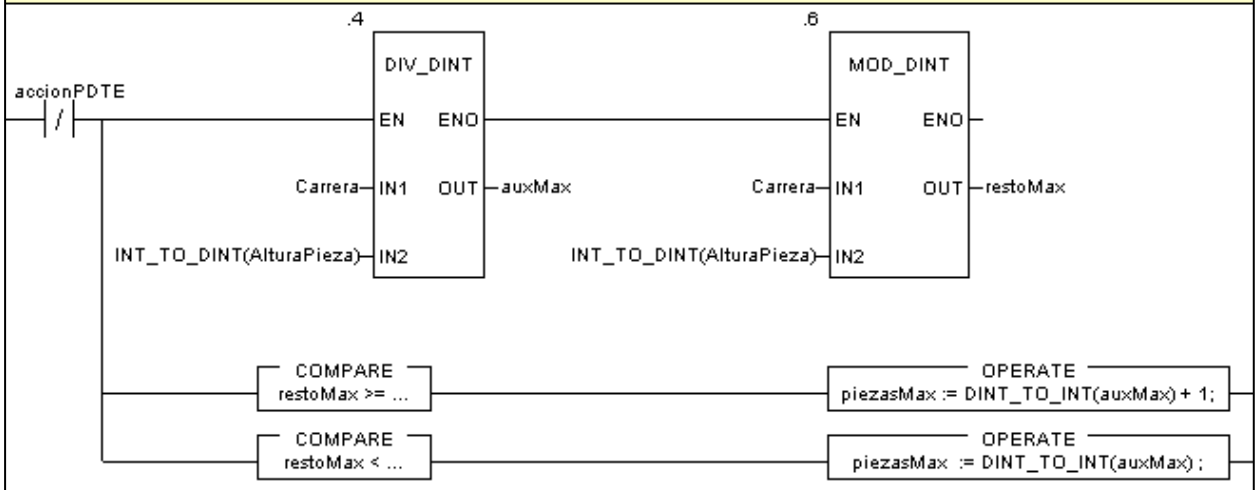
Esta sección comenzara a ejecutarse cuando se indique desde su sección controladora, la cual reestablecera a false la posición de HOME. La sección de DISPOSITIVO sera la encargada de la busqueda de HOME. Esta sección actuará tras detectarse HOME, indicando el desplazamiento hacia posicion superior. Una vez alcanzada se activará el sensor de piezas (false), pararemos el sistema tomando la posición como referencia para futuros desplazamientos y calcularemos el numero de piezas en la maquina



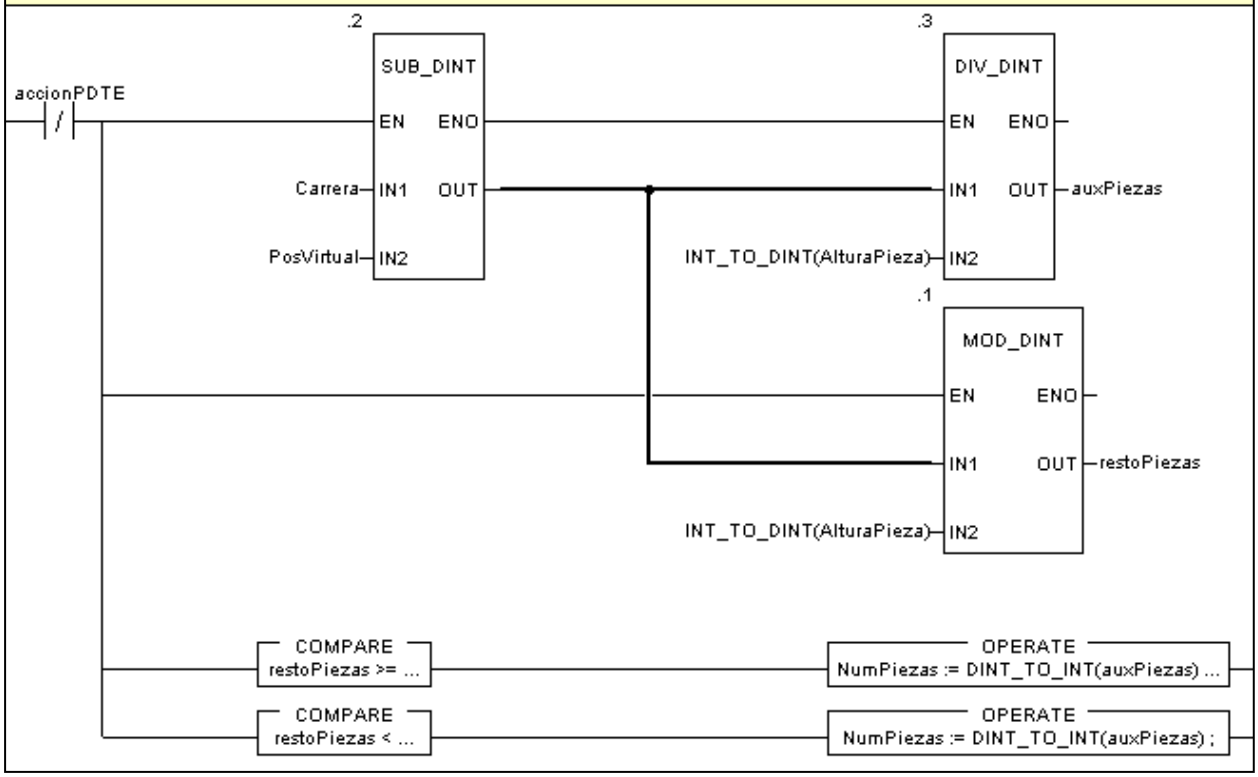
Esta seccion contabiliza las piezas que se introducen/extran de la maquina. El criterio usado es que tanto cuando se acabe de realizar el test como cuando se esté esperando para procesar peticiones se actualizará el valor del numero de piezas. Realmente lo que se hace es ver cuanto espacio no se recorre y con eso hallar las piezas en el dispensador.

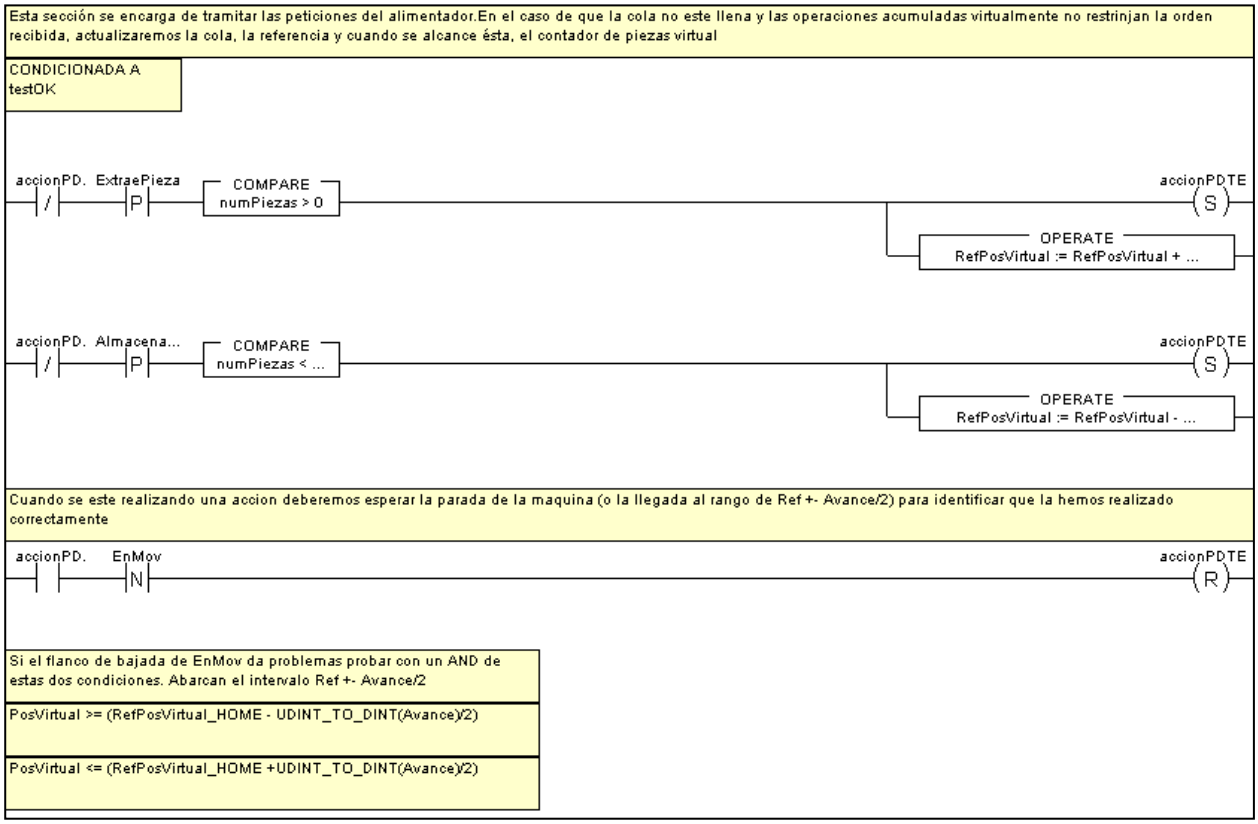
CONDICIONADA A testOK	La division entre enteros TRUNCA hacia abajo

Siempre y cuando el test se haya realizado corectamente y no haya acciones pendientes se calcula el numero maximo de piezas del alimentador

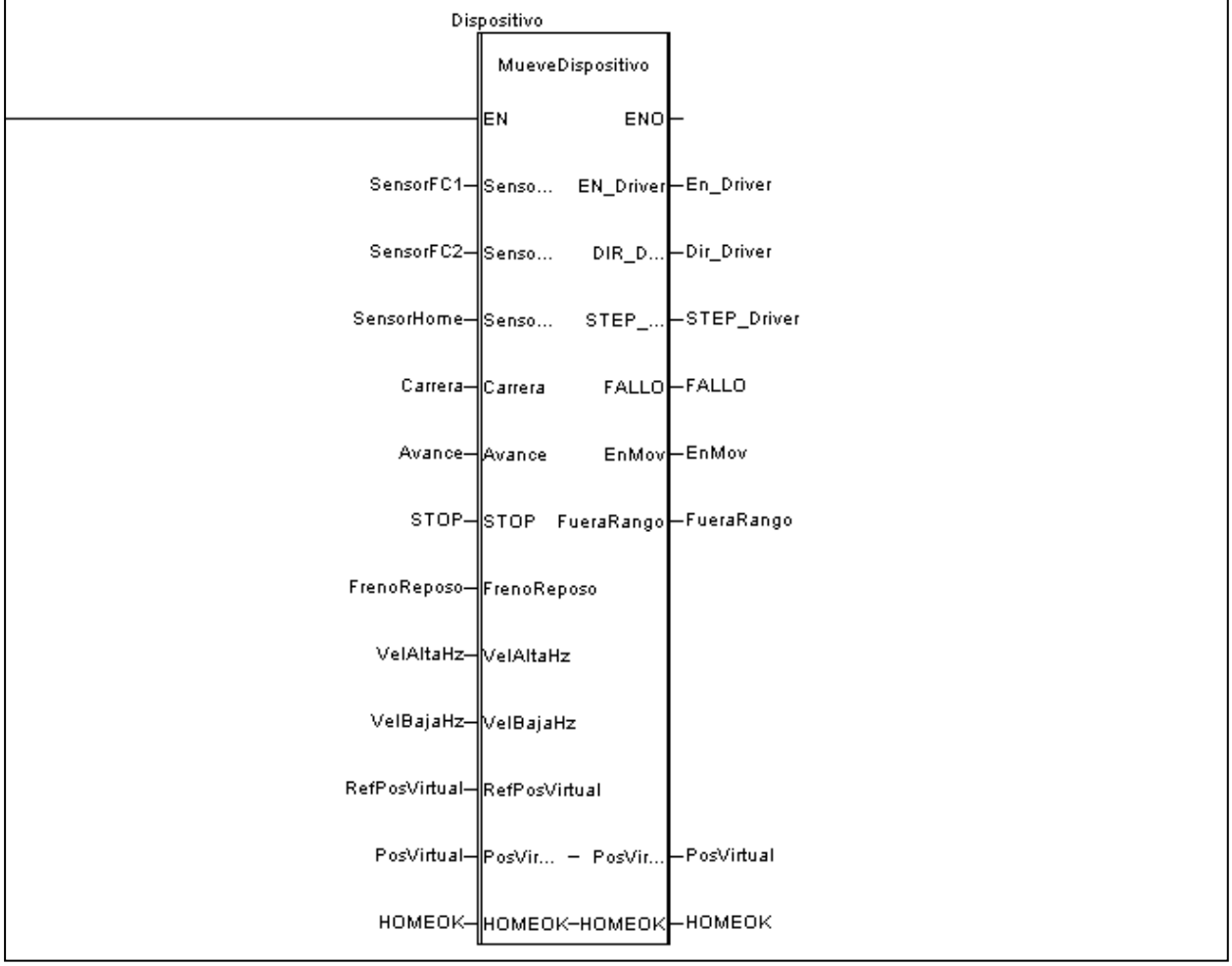


Siempre y cuando el test se haya realizado corectamente y no haya acciones pendientes se actualizará el valor del numero de piezas en el alimentador de forma continua. Para ello se redondeara

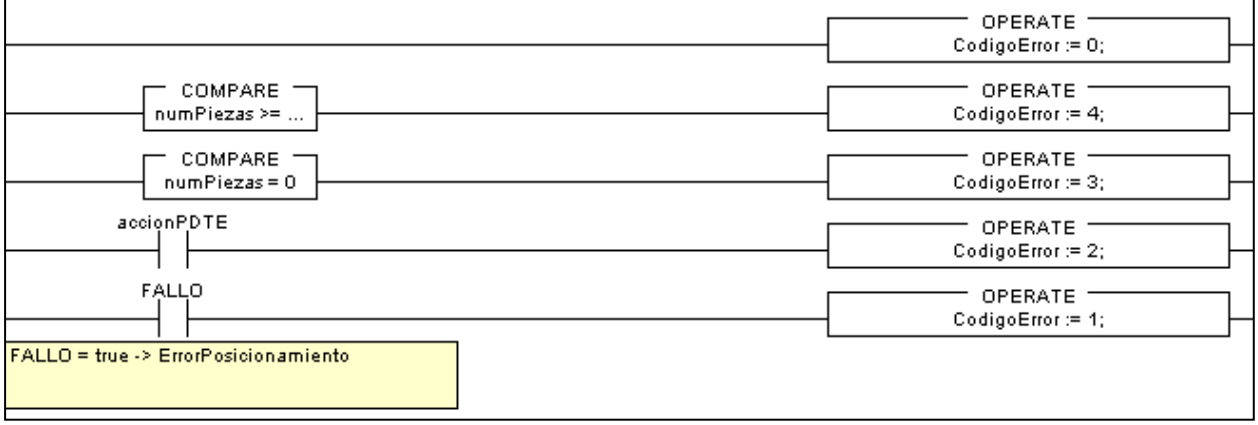



Esta sección se encarga de tramitar las peticiones del alimentador.En el caso de que la cola no este llena y las operaciones acumuladas virtualmente no restrinjan la orden recibida, actualizaremos la cola, la referencia y cuando se alcance ésta, el contador de piezas virtual
CONDICIONADA A testOK
accionPD. ExtraePieza
P
COMPARE numPiezas > 0
accionPDTE
S
OPERATE RefPosVirtual := RefPosVirtual + ...
accionPD. Almacena...
P
COMPARE numPiezas < ...
accionPDTE
S
OPERATE RefPosVirtual := RefPosVirtual - ...
Cuando se este realizando una accion deberemos esperar la parada de la maquina (o la llegada al rango de Ref +- Avance/2) para identificar que la hemos realizado correctamente
accionPD.
EnMov
N
accionPDTE
R
Si el flanco de bajada de EnMov da problemas probar con un AND de estas dos condiciones. Abarcan el intervalo Ref +- Avance/2
PosVirtual >= (RefPosVirtual_HOME - UDINT_TO_DINT(Avance)/2)
PosVirtual <= (RefPosVirtual_HOME +UDINT_TO_DINT(Avance)/2)

Esta sección está encargada de la función de desplazamiento del Dispositivo



Esta sección se encarga de gestionar la posible problematica y transmitir las situaciones mediante códigos de errores



9 ANEXOS

9.1 Lista de señales y numeración bornero

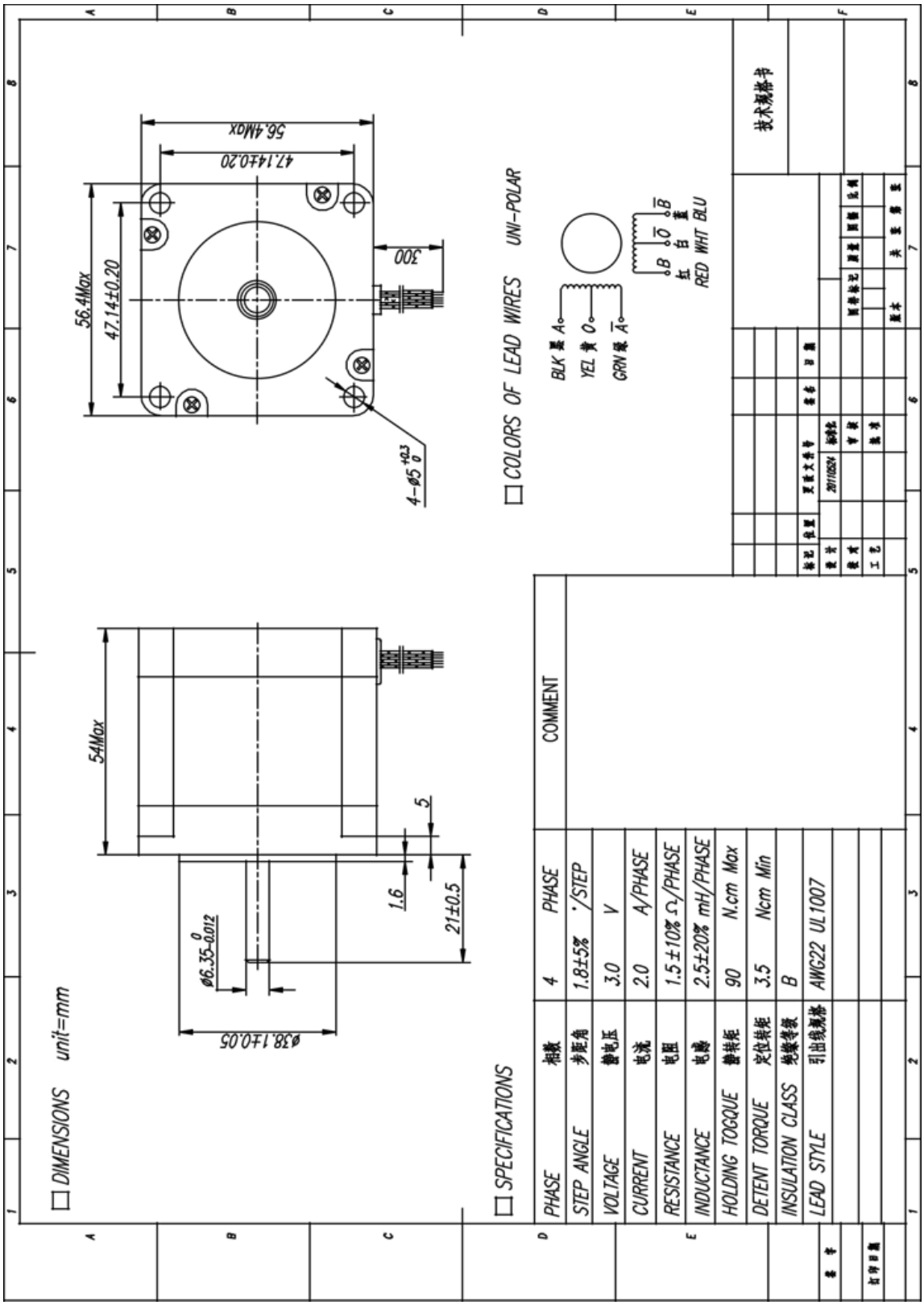
Lista de señales comprende los dos alimentadores del dispositivo.

	Nombre	Entrada	Manguera	Color	M340
NO	ED_ALI_PIEZ1_HOME	Bl	BB (entrada)	Ma	%I0.1.49
NC	ED_ALI_PIEZ1_FOTO	Ne	BB (entrada)	Ve	%I0.1.50
NO	ED_ALI_PIEZ2_HOME	Rs	BB (entrada)	Am	%I0.1.51
NC	ED_ALI_PIEZ2_FOTO	Az	BB (entrada)	Gr	%I0.1.52
24V	**SD_ALI_PIEZ1_PUL**	Am	BA (salida)	Bl/Am	%Q0.2.46
24V	**SD_ALI_PIEZ1_DIR**	Vi	BA (salida)	Am/Ma	%Q0.2.47
24V	**SD_ALI_PIEZ1_EN**	Ro	BB (salida)	Bl	%Q0.2.48
24V	**SD_ALI_PIEZ2_ PUL**	Gr	BB (salida)	Ma	%Q0.2.49
24V	**SD_ALI_PIEZ2_ DIR**	Ma	BB (salida)	Ve	%Q0.2.50
24V	**SD_ALI_PIEZ2_EN**	Ve	BB (salida)	Am	%Q0.2.51

Mang. PLC			MAQUINA			
Ro			1	+24V	Ro	
Az			2	-0v	Az	
Ma	R2K		3	DIR+ (Driver Der)	Bl	SD_ALI_PIEZ2_DIR
Gr	R2K		4	PULSO+ (Driver Der)	Gr	SD_ALI_PIEZ2_PUL
			5	-5V CC (Driver Der)	Az	
Ve	R2K		6	ENABLE + (Driver Der)	Ve	SD_ALI_PIEZ2_EN
			7	(pulsador Der) conecta Na con Bl	Na	
			8	(pulsador Der) conecta Am con Gr	Am	
			9	+5Vcc para dos pulsadores	Ro	
			10	Generador de pulsos	Ne	
			11	FC (seguridad) IN 4 fines en serie		
			12	FC (seguridad) OUT 4 fines serie		
Vi	R2K		13	DIR+ (Driver Izq)	Bl	SD_ALI_PIEZ1_DIR
Am	R2K		14	PULSO+ (Driver Izq)	Gr	SD_ALI_PIEZ1_PUL
Ro	R2K		15	ENABLE + (Driver Izq)	Ve	SD_ALI_PIEZ1_EN
			16	(pulsador Izq) conecta Ve con Bl	Ve	
			17	(pulsador Izq) conecta Bl con Gr	Bl	
			18	Generador de pulsos	Gr	
Rs			19	Sensor de cero (Der) home	Ne	ED_ALI_PIEZ2_HOME
Bl			20	Sensor de Cero (Izq) Home	Bl	ED_ALI_PIEZ1_HOME
-Ro			21	+24V	Ro	
-Az			22	-0V	Az	
Az			23	FotoCelula Barrera (DER) señal	Ne	ED_ALI_PIEZ2_FOTO
			24	FotoCelula Barrera (DER) función luz-sombra	Bl	
			25	Emisor FotoCelula Barrera (DER) TEST+	Am	
			26	Emisor FotoCelula Barrera (DER) TEST-	Ve	
Ne			27	FotoCelula Barrera (IZQ) señal	Ne	ED_ALI_PIEZ1_FOTO
			28	FotoCelula Barrera (IZQ) función luz-sombra	Bl	
			29	Emisor FotoCelula Barrera (IZQ) TEST+	Am	
			30	Emisor FotoCelula Barrera (IZQ) TEST-	Ve	

9.2 Datasheets

9.2.1 Motor



9.2.2 Driver

ECG Safety Statement

Easy Commercial Global is not liable or responsible for any accidents, injuries, equipment damage, property damage, loss of money or loss of time resulting from improper use of electrical or mechanical or software products sold on this website or other Easy Commercial Global sales resources.

Since Easy Commercial Global basically provide OEM machine builders components to build their machines for their own use or third party use it is their responsibility to maintain certify and comply the end user products built base on our components sold on this website or other Easy Commercial Global sales resources.

Assembling electrical CNC machine component like power supplies, motors, drivers or other electrical components involve dealing with high voltage like AC alternative current or DC direct current which is extremely dangerous and need high attention & essential experience and knowledge of software, electricity, electro-mechanics & or mechanics.

For technical questions please contact us at ebay@savebase.com before purchase.

1. Introduction, Features and Applications

Introduction

The TB1H is a high performance microstepping driver based on the latest original TOSHIBA high-efficiency TB6600HG IC. The TB6600HG adopts PWM chopper-type single-chip bipolar sinusoidal to ensure the low vibration and high efficiency. Moreover, the brand new design with BiCD0.13 (50V) process technology on the chipset also ensures maximum 5.0A output current and 50V output withstand voltage. Consequently, as long as the current range of the stepper motor is within 0.2-5 amps, all the 2 Phase or 4 Phase of Nema17, Nema23, Nema24 and Nema34 stepper motors will work perfectly with this new-type TB6600HG Stepper Driver.

Features

- High performance, cost-effective
- Automatic idle-current reduction
- Supply voltage up to 50V DC
- Output current up to 5.0A
- Suitable for 2-phase and 4-phase motors
- High speed optoelectronic isolation signal input
- Overload, overcurrent, overheat, overvoltage and undervoltage protection
- Single-chip PWM bipolar sinusoidal chopper ensures low vibration and high efficiency
- 1, 2, 4 (**New Mode**), 8, 16 adjustable microstep control, motors run more precisely and smoothly
- Equipped with the 3rd generation of breakout board, display panel and control pad to control the motor manually.
- Cooling Aluminium Box Design for Cooling, and protect the driver board from being damaged by dirt, dust or other liquids.

Applications

Suitable for a wide range of stepping motors, from NEMA size 17 to 34. It can be used in various kinds of machines, such as X-Y tables, labeling machines, laser cutters, engraving machines, pick-place devices, and so on. Particularly adapt to the applications desired with low noise, low heating, and high speed performance.

2. Specifications

I. Electrical Specifications (T_j = 25℃/77℉)

Parameters	TB1H			
	Min	Typical	Max	Unit
Output current	1	-	5.0(3 RMS)	A
Supply voltage	+12	+24	+50	VDC
Logic signal current	7	10	16	mA
Pulse input frequency	0	-	300	KHz
Isolation resistance	500			MΩ

II. Operating Environment and other Specifications

Cooling	Natural Cooling or Forced cooling	
Operating Environment	Environment	Avoid dust, oil fog and corrosive gases
	Ambient Temperature	0 ℃ — 50℃ (32℉ — 122℉)
	Humidity	40%RH — 90%RH
	Operating Temperature	70℃ (158℉) Max
	Vibration	5.9m/s2 Max
Storage Temperature	-20 ℃ — 65℃ (-4℉ — 149℉)	
Weight	Approx. 250g	

III. PCB Instructions & Specifications (unit: mm)

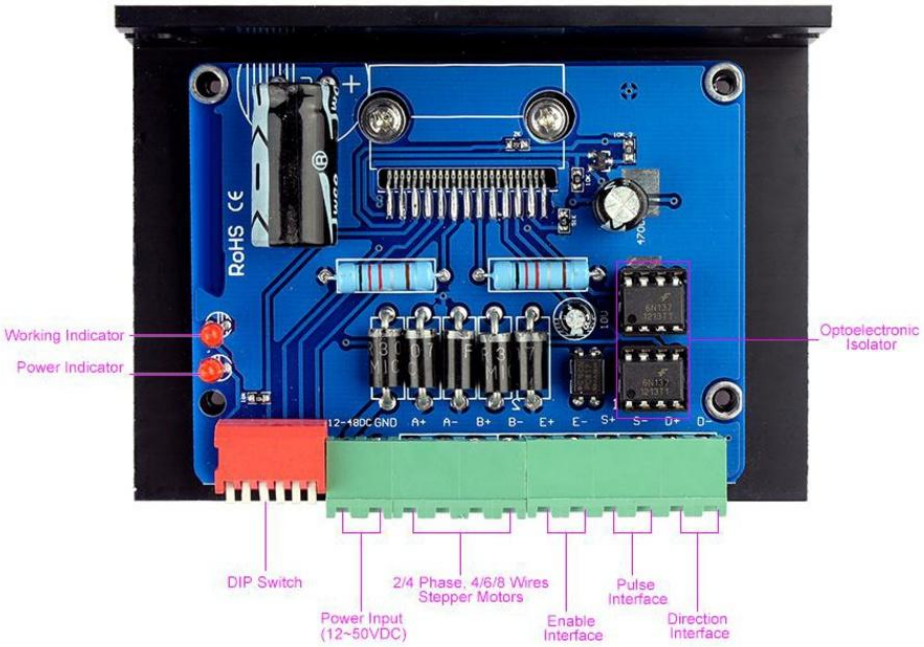


Figure 1: PCB Instructions & Specifications.

3. Pin Assignment & Instructions

I. Control Signal Instructions:

Pin Function	Details
PUL+ / PUL-	PUL+: Step pulse signal input positive terminal PUL-: Step pulse signal input negative terminal.
DIR+ / DIR-	PUL+: Step direction signal input positive terminal PUL-: Step direction signal input negative terminal. Please note that motion direction is also related to motor-driver wiring match. Exchanging the connection of two wires for a coil to the driver will reverse motion direction.
ENA+ / ENA-	Enable signal: This signal is used for enabling/disabling the driver. High level (NPN control signal, PNP and Differential control signals are on the contrary, namely Low level for enabling.) for enabling the driver and low level for disabling the driver. Usually left **UNCONNECTED (ENABLED)**

II. Power Input Terminals & Motor connection Terminals

Pin Function	Details
DC+ / DC-	Switching power supply, 12~48 VDC.
A+ / A-	Motor Phase A
B+ / B-	Motor Phase B

4. NPN, PNP Wiring Diagram for reference

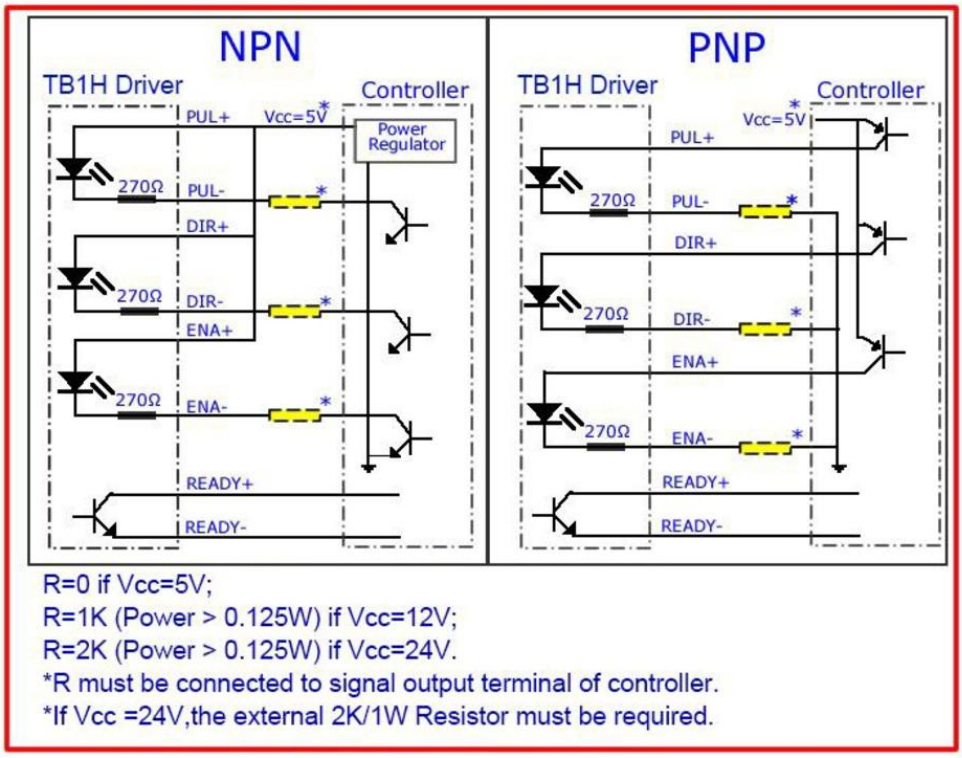


Figure 2: NPN/PNP Wiring Diagram

5. Selections & Connections about the Motors

The TB1H stepper driver can drive 2-pahse and 4-pahse hybrid stepping motors, including 4, 6 or 8 leads.

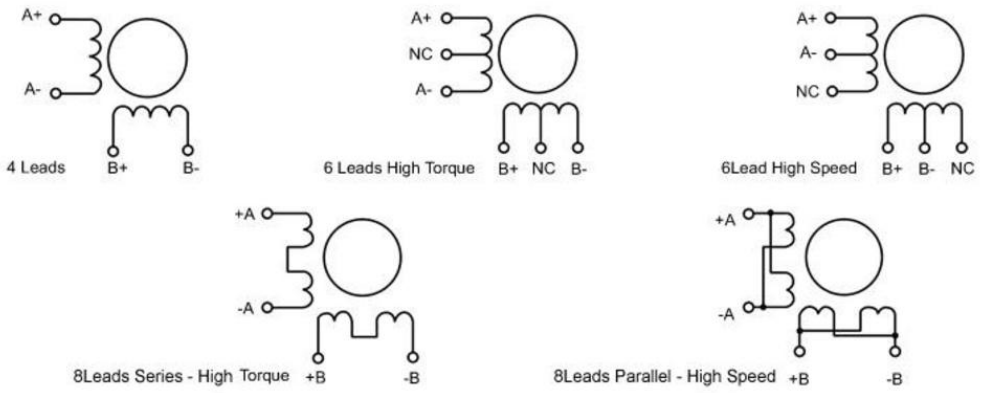


Figure 3: Wiring diagrams for 4/6/8 leads motors

I. Connections of 4-lead Motors

4 lead motors are the least flexible but easiest to wire. Speed and torque will depend on winding inductance. In theory, during adjusting stepper driver's output current, the output current can be set to 1.4 times than the rated current of the motor on the premise that the 1.4 times of rated current is lower than the TB6600HG chip's 5.0A peak current.

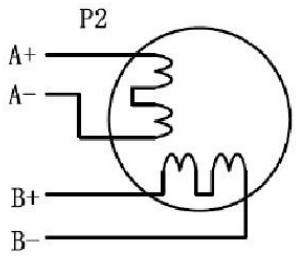


Figure 4: 4-lead Motor Connections

II. Connections of 6-lead Motors

Like 8 lead stepping motors, 6 lead motors have two configurations available for high speed or high torque operation. The higher speed configuration, or half coil, is so described because it uses one half of the motor's inductor windings. The higher torque configuration, or full coil, uses the full windings of the phases.

i. Half Coil Configurations

As previously stated, the half coil configuration uses 50% of the motor phase windings. This gives lower inductance, hence, lower torque output. Like the parallel connection of 8 lead motor, the torque output will be more stable at higher speeds. This configuration is also referred to as half chopper. In setting the driver output current multiply the specified per phase (or unipolar) current rating by 1.4 to determine the peak output current.

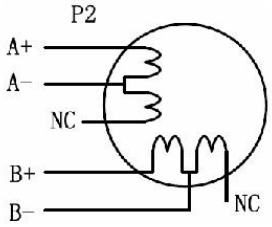


Figure 5: 6-lead motor half coil (higher speed) connections

ii. Full Coil Configurations

The full coil configuration on a six lead motor should be used in applications where higher torque at lower speeds is desired. This configuration is also referred to as full copper. In full coil mode, the motors should be run at only 70% of their rated current to prevent overheating.

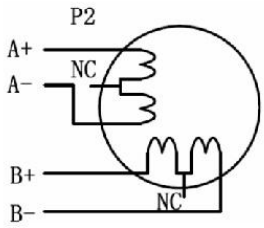


Figure 6: 6-lead motor full coil (higher torque) connections

III. Connections of 8-lead Motors

8 lead motors offer a high degree of flexibility to the system designer in that they may be connected in series or parallel, thus satisfying a wide range of applications.

i. Series Connections

A series motor configuration would typically be used in applications where a higher torque at lower speeds is required. Because this configuration has the most inductance, the performance will start to degrade at higher speeds. In series mode, the motors should also be run at only 70% of their rated current to prevent overheating.

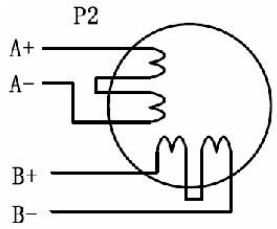


Figure 7: 8-lead motor series (higher torque) connections

ii. Parallel Connections

An 8 lead motor in a parallel configuration offers a more stable, but lower torque at lower speeds. But because of the lower inductance, there will be higher torque at higher speeds. Multiply per phase (or unipolar) current rating by 1.96, or the bipolar current rating by 1.4, to determine the peak output current.

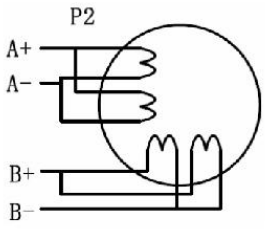


Figure 8: 8-lead motor parallel (higher speed) connections

6. Power Supply Selection

The TB1H stepper driver can match Large and small size stepping motors (from Nema size 17 to 34) made by us or other motor manufactures around the world, as long as the rated current of the motors is within **0.2-5.0A**(Peak Current) . To achieve good driving performances, it is important to select supply voltage and output current properly. Generally speaking, supply voltage determines the high speed performance of the motor, while output current determines the output torque of the driven motor (particularly at lower speed). Higher supply voltage will allow higher motor speed to be achieved, at the price of more noise and heating. If the motion speed requirement is low, it's better to use lower supply voltage to decrease noise, heating and improve reliability.

I. Regulated or Unregulated Power Supply

Both of regulated and unregulated DC power supplies can be used to supply TB1H stepper driver. However, unregulated power supplies are preferred due to their ability to withstand current surge. If regulated power supplies (such as most off switching supplies.) are indeed used, it is important to have large current output rating to avoid problems like current clamp, for example using 4A supply for 3A motor-driver operation. On the other hand, if unregulated supply is used, one may use a power supply of lower current rating than that of motor (typically 50%～70% of motor current). The reason is that the driver draws current from the power supply capacitor of the unregulated supply only during the ON duration of the PWM cycle, but not during the OFF duration. Therefore, the average current withdrawn from power supply is considerably less than motor current. For example, two 3A motors can be well supplied by one power supply of 4A rating. Although the unregulated power supplies are preferred, considering the cost, the cheap and easy-to-use regulated switching supplies in the market is also a good choice for the TB1H stepper driver and motors, as long as the total output current of the regulated switching supplies is larger than the motor's total rated current. Anyway, if users don't know how to select the suitable power supplies for the TB1H stepper driver and motors, please feel free to contact with us for assistances.

II. Selecting Supply Voltage

The TB1H stepper driver can actually operate within **12-50V**(Peak Voltage) DC for different motors. Higher supply voltage can increase motor torque at higher speeds, thus helpful for avoiding losing steps. However, higher voltage may cause more motor vibration at lower speed, and it may also cause motor overheat and drive damage. Therefore, it is suggested to choose 12-24V DC supply to power the Nema17 motors, 24-36V DC supply to power the Nema23/24 motors and 36V-48V DC supply to power the Nema34 Motors.

7. DIP Switches Settings (Microstep, Current)

This driver adopts a 6-bit DIP switch to set motor operating current and microstep mode, shown as below:

Figure 9: DIP Switches Settings

I. Microstep Mode Selection:

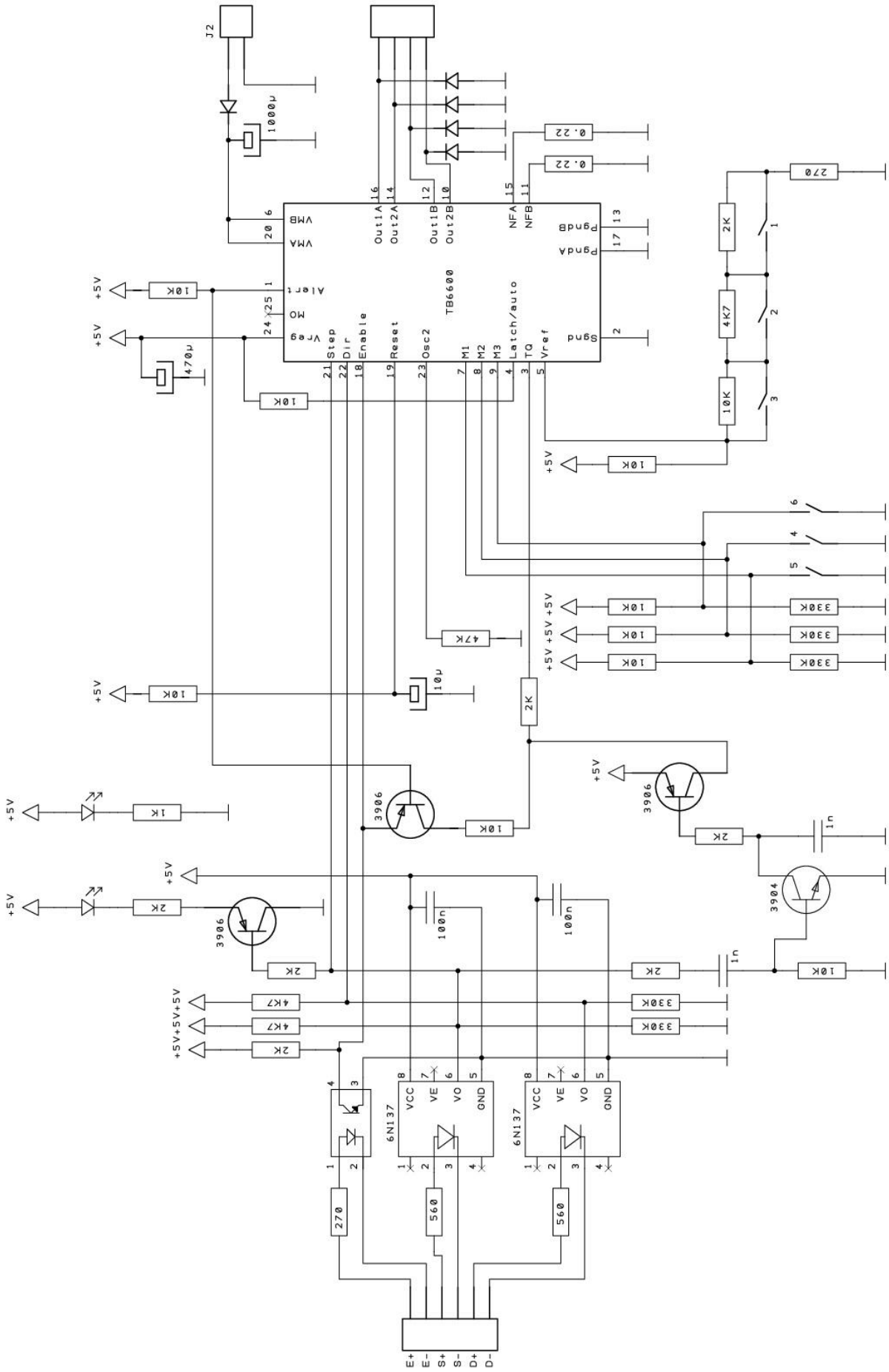
Microstep Mode	SW1	SW2	SW3
N/A	ON	ON	ON
1(Full Step)	OFF	ON	ON
2	ON	OFF	ON
2	OFF	OFF	ON
4	ON	ON	OFF
8	OFF	ON	OFF
16	ON	OFF	OFF
N/A	OFF	OFF	OFF

II. Output Current Selection:

Output Current	SW4	SW5	SW6
0.2A	ON	ON	ON
0.6A	OFF	ON	ON
1.2A	ON	OFF	ON
1.8A	OFF	OFF	ON
2.5A	ON	ON	OFF
3.3A	OFF	ON	OFF
4.2A	ON	OFF	OFF
5.0A	OFF	OFF	OFF

8. Wiring Notes

- In order to improve anti-interference performance of the driver, it is recommended to use twisted pair shield cable.
- Please shut down the power before plugging or unplugging the connectors from the driver.


J2
1000µ
TB6600
VMB
VMA
Alert
MO
Vreg
Step
Dir
Enable
Reset
Osc2
M1
M2
M3
Latch/auto
TQ
Vref
Gnd
PgndA
PgndB
Out1A
Out2A
Out1B
Out2B
NFA
NFB
0.22
0.22
270
2K
4K7
10K
+5V
10K
470µ
10µ
47K
330K
3906
3904
1K
2K
100n
1n
4K7
6N137
VCC
VE
VO
GND
560
E+
E-
S+
S-
D+
D-

TOSHIBA TB6600HQ

Preliminary

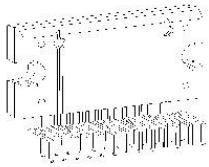
TOSHIBA BiCD Integrated Circuit Silicon Monolithic

TB6600HQ

PWM Chopper-Type bipolar
Stepping Motor Driver IC

The TB6600HQ is a PWM chopper-type single-chip bipolar sinusoidal micro-step stepping motor driver.
Forward and reverse rotation control is available with 2-phase, 1-2-phase, W1-2-phase, 2W1-2-phase, and 4W1-2-phase excitation modes.
2-phase bipolar-type stepping motor can be driven by only clock signal with low vibration and high efficiency.

TB6600HQ



HZIP25-1.00F

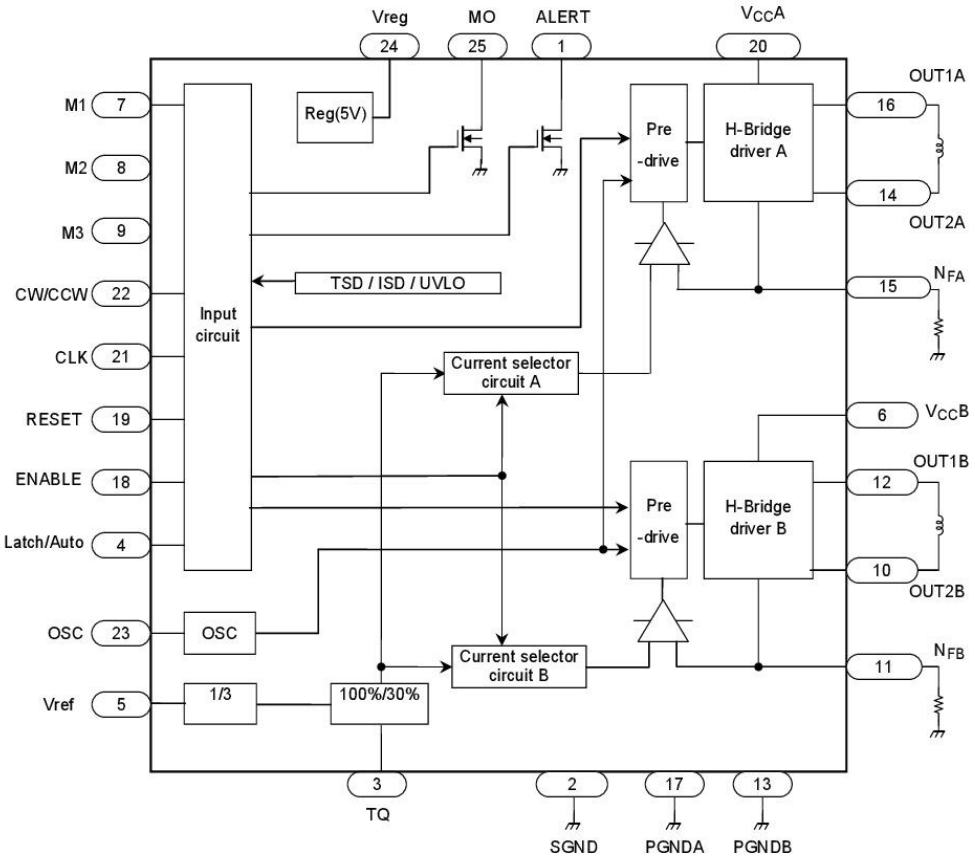
Weight:
HZIP25-1.00F: 7.7g (typ.)

Features

- Single-chip bipolar sinusoidal micro-step stepping motor driver
- BiCD 0.13 (50 V) process
- Ron (upper + lower) = 0.4 Ω (typ.)
- Forward and reverse rotation control available
- Selectable phase drive (1/1, 1/2, 1/4, 1/8, and 1/16 step)
- Output withstand voltage: V_{CC} = 50 V
- Output current: I_{OUT} = 5.0 A (absolute maximum ratings, peak, within 100ms)
  I_{OUT} = 4.5 A (operating range, maximal value)
- Packages: HZIP25-1.00F
- Built-in input pull-down resistance: 100 kΩ (typ.)
- Output monitor pins (ALERT): Maximum of I_{ALERT} = 1 mA
- Output monitor pins (MO): Maximum of I_{MO} = 1 mA
- Equipped with reset and enable pins
- Stand by function
- Single power supply
- Built-in thermal shutdown (TSD) circuit
- Built-in under voltage lock out (UVLO) circuit
- Built-in over-current detection (ISD) circuit

The following conditions apply to solderability:
About solderability, following conditions were confirmed
(1)Use of Sn-37Pb solder Bath
·solder bath temperature: 230℃·dipping time: 5 seconds·the number of times: once·use of R-type flux
(2)Use of Sn-3.0Ag-0.5Cu solder Bath
·solder bath temperature: 245℃·dipping time: 5 seconds·the number of times: once·use of R-type flux

Block Diagram



Setting of Vref

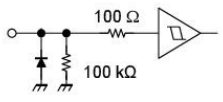
Input TQ	Voltage ratio
L	30%
H	100%

Pin Functions

Pin No.	I/O	Symbol	Functional Description	Remark
1	Output	ALERT	TSD / ISD monitor pin	Pull-up by external resistance
2	—	SGND	Signal ground	
3	Input	TQ	Torque (output current) setting input pin	
4	Input	Latch/Auto	Select a return type for TSD and ISD.	L: Latch, H: Automatic return
5	Input	Vref	Voltage input for 100% current level	
6	Input	VccB	B channel Power supply	
7	Input	M1	Excitation mode setting input pin	
8	Input	M2	Excitation mode setting input pin	
9	Input	M3	Excitation mode setting input pin	
10	Output	OUT2B	B channel output 2	
11	—	N_{FB}	B channel output current detection pin	
12	Output	OUT1B	B channel output 1	
13	—	PGNDB	Power ground	
14	Output	OUT2A	A channel output 2	
15	—	N_{FA}	A channel output current detection pin	
16	Output	OUT1A	A channel output 1	
17	—	PGNDA	Power ground	
18	Input	ENABLE	Enable signal input pin	H: Enable, L: All outputs off
19	Input	RESET	Reset signal input pin	L: Initial mode
20	Input	VccA	A channel Power supply	
21	Input	CLK	CLK pulse input pin	
22	Input	CW/CCW	Forward/reverse control pin	L: CW, H:CCW
23	—	OSC	Resistor connection pin for internal oscillation setting	
24	Output	V_{reg}	Control side connection pin for power capacitor	Connecting capacitor to SGND
25	Output	MO	Electrical angle monitor pin	Pull-up by external resistance

<Terminal circuits>

Input pins
(M1, M2, M3,CLK, CW/CCW,
ENABLE, RESET, TQ, Latch/Auto)



Absolute Maximum Ratings (Ta = 25°C)

Characteristic	Symbol	Rating	Unit
Power supply voltage	V_{CC}	50	V
Output current (per one phase)	$I_{O\ (PEAK)}$	5.0/phase (Note 1)	A
Drain current (ALERT, DOWN)	$I_{(ALERT)}$	1	mA
	$I_{(MO)}$		
Input voltage	V_{IN}	6	V
Power dissipation	P_D	3.2 (Note 2)	W
		40 (Note 3)	
Operating temperature	T_{opr}	−30 to 85	°C
Storage temperature	T_{stg}	−55 to 150	°C

Note 1: T = 100ms

Note 2: Ta = 25°C, No heat sink

Note 3: Ta = 25°C, with infinite heat sink.

The absolute maximum ratings of a semiconductor device are a set of ratings that must not be exceeded, even for a moment. Do not exceed any of these ratings.
Exceeding the rating (s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.
Please use the IC within the specified operating ranges.

Operating Range (Ta = 25°C)

Characteristic	Symbol	Test Condition	Min	Typ.	Max	Unit
Power supply voltage	V_{CC}	—	8.0	—	45	V
Output current	I_{OUT}	—	—	—	4.5	A
Input voltage	V_{IN}	—	0	—	5.5	V
	V_{ref}	—	0.3	—	3.5	V
Clock frequency	f_{CLK}	—	—	—	200	kHz
Chopping frequency	f_{chop}	(kΩ) ≦ Rosc ≦ (kΩ)	20	40	60	kHz
OSC frequency	f_{OSC}	R_{OSC} = 51 kΩ	(2.0)	4.0	(6.0)	MHz

Note: $V_{CC}A$ and $V_{CC}B$ should be programmed the same voltage.
The maximum current of the operating range can not be necessarily conducted depending on various conditions because output current is limited by the power dissipation Pd.
Make sure to avoid using the IC in the condition that would cause the temperature to exceed Tj (avg.) (107°C).

Electrical Characteristics (Ta = 25°C, V_{CC} = 24 V)

Characteristic		Symbol	Test Condition	Min	Typ.	Max	Unit
Input voltage	High	$V_{IN\ (H)}$	M1, M2, M3, CW/CCW, CLK, RESET, ENABLE, Latch/Auto, TQ	2.0	—	5.5	V
	Low	$V_{IN\ (L)}$		−0.2	—	0.8	
Input hysteresis voltage		V_H		—	400	—	mV
Input current		$I_{IN\ (H)}$	M1, M2, M3, CW/CCW, CLK, RESET, ENABLE, Latch/Auto, TQ V_{IN} = 5.0 V	—	55	80	μA
		$I_{IN\ (L)}$	V_{IN} = 0 V	—	—	1	
V_{CC} supply current		Icc_1	Output open, RESET: H, ENABLE: H、M1:L, M2:L, M3:H (1/1-step mode) CLK:L	—	3.1	(7)	mA
		Icc_2	Output open, RESET: L, ENABLE: L M1:L, M2:L, M3:H (1/1-step mode) CLK:L	—	3.1	(7)	
		Icc_3	Standby mode (M1:L, M2:L, M3:L)	—	1.8	(4)	
Vref input circuit	Current limit voltage	V_{ref}	Vref = 3.0 V	0.9	1.0	1.1	V
	Input current	$I_{IN(ref)}$	Vref = 3.0 V	—	—	1	μA
	Divider ratio	V_{ref}/V_{NF}	Maximum current : 100%	—	3	—	—
Minimum CLK pulse width		tw_{CLKH}		2.2	—	—	μs
		tw_{CLKL}					
Output residual voltage		V_{OL} MO	I_{OL} = 1 mA	—	—	0.5	V
		V_{OL} ALERT					
Internal constant voltage		Vreg	External capacitor = 0.1 μF (in standby mode)	4.5	5.0	5.5	V
TSD operation temperature (Note)		TSD	Design target value	160	170	180	°C
TSD hysteresis(Note)		TSDhys	Design target value	30	40	50	°C
Over current detection current (Note)		ISD	All outputs, Design target value	5.0	6.5	8.0	A
Oscillation frequency		f_{OSC}	External resistance Rosc = 51 kΩ	2.8	4	5.2	MHz

Note: Pre-shipment testing is not performed.

Electrical Characteristics (Ta = 25°C, V_{CC} = 24 V)

Characteristic		Symbol	Test Condition	Min	Typ.	Max	Unit
Output ON resistor		$Ron_U + Ron_L$	I_{OUT} = 4 A	—	0.4	0.6	Ω
Output transistor switching characteristics		t_r	V_{NF} = 0 V, Output: Open	—	(0.5)	—	μs
		t_f		—	(0.5)	—	
Output leakage current	Upper side	I_{LH}	V_{CC} = 50 V	—	—	5	μA
	Lower side	I_{LL}		—	—	5	

Description of Functions

1. Excitation Settings

The excitation mode can be selected from the following eight modes using the M1, M2 and M3 inputs. New excitation mode starts from the initial mode when M1, M2, or M3 inputs are shifted during motor operation. In this case, output current waveform may not continue.

Input			Mode (Excitation)
M1	M2	M3	
L	L	L	Standby mode (Operation of the internal circuit is almost turned off.)
L	L	H	1/1 (2-phase excitation, full-step)
L	H	L	1/2A type (1-2 phase excitation A type) (0% - 71% - 100%)
L	H	H	1/2B type (1-2 phase excitation B type) (0% - 100%)
H	L	L	1/4 (W1-2 phase excitation)
H	L	H	1/8 (2W1-2 phase excitation)
H	H	L	1/16 (4W1-2 phase excitation)
H	H	H	Standby mode (Operation of the internal circuit is almost turned off.)

Note: To change the exciting mode by changing M1, M2, and M3, make sure not to set M1 = M2 = M3 = L or M1 = M2 = M3 = H.

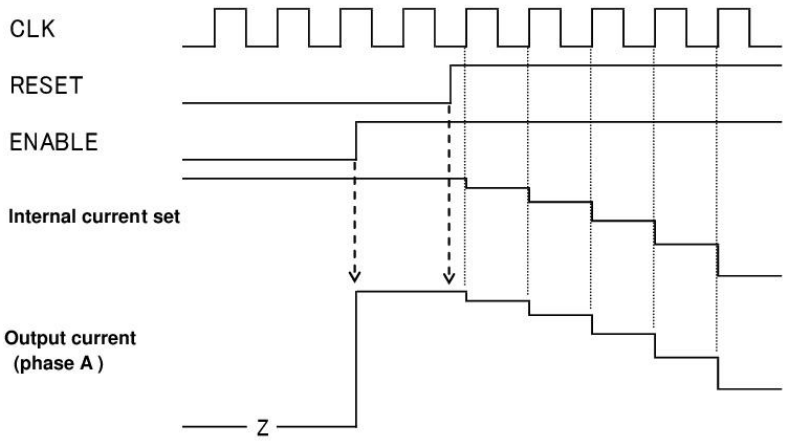
Standby mode

The operation mode moves to the standby mode under the condition M1 = M2 = M3 = L or M1 = M2 = M3 = H.
The power consumption is minimized by turning off all the operations except protecting operation.
In standby mode, output terminal MO is HZ.
To release the standby mode, release the condition of M1 = M2 = M3 = L or M1 = M2 = M3 = H.
Input signal is not accepted for about 200 μs after releasing the standby mode.

2. Function

(1) When the ENABLE signal goes Low level, it sets an OFF on the output.
(2) The output changes to the Initial mode shown in the table below when the ENABLE signal goes High level and the RESET signal goes Low level. In this mode, the status of the CLK and CW/CCW pins are irrelevant.
(3) When the ENABLE signal goes Low level, it sets an OFF on the output. In this mode, the output changes to the initial mode when the RESET signal goes Low level. Under this condition, the initial mode is output by setting the ENABLE signal High level. And the motor operates from the initial mode by setting the RESET signal High level.

(Example 1)



Input				Output mode
CLK	CW/CCW	RESET	ENABLE	
↑	L	H	H	CW
↑	H	H	H	CCW
X	X	L	H	Initial mode
X	X	X	L	Z

X: Don't Care

Command of the standby has a higher priority than ENABLE. Standby mode can be tured on and off regardless of the state of ENABLE.

3. Initial Mode

When RESET is used, the phase currents are as follows.

Excitation Mode	Phase A Current	Phase B Current
1/1 (2-phase excitation, full-step)	100%	-100%
1/2A type (1-2 phase excitation A type) (0% - 71% - 100%)	100%	0%
1/2B type (1-2 phase excitation B type) (0% - 100%)	100%	0%
1/4 (W1-2 phase excitation)	100%	0%
1/8 (2W1-2 phase excitation)	100%	0%
1/16 (4W1-2 phase excitation)	100%	0%

In this specification, current direction is defined as follows.
OUT1A → OUT2A: Forward direction
OUT1B → OUT2B: Forward direction

4. 100% current settings (Current value)

100% current value is determined by Vref inputted from external part and the external resistance for detecting output current. Vref is doubled 1/3 inside IC.

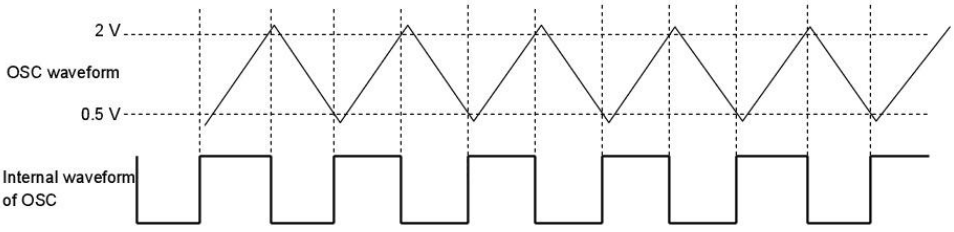
$$Io(100\%) = (1/3 \times Vref) \div R_{NF}$$

The average current is lower than the calculated value because this IC has the method of peak current detection.
R_{NF} should be 0.2 Ω or more.

5. OSC

Triangle wave is generated internally by CR oscillation by connecting external resistor to OSC terminal.
Rosc: (kΩ) ≤ Rosc ≤ (kΩ)

OSC waveform

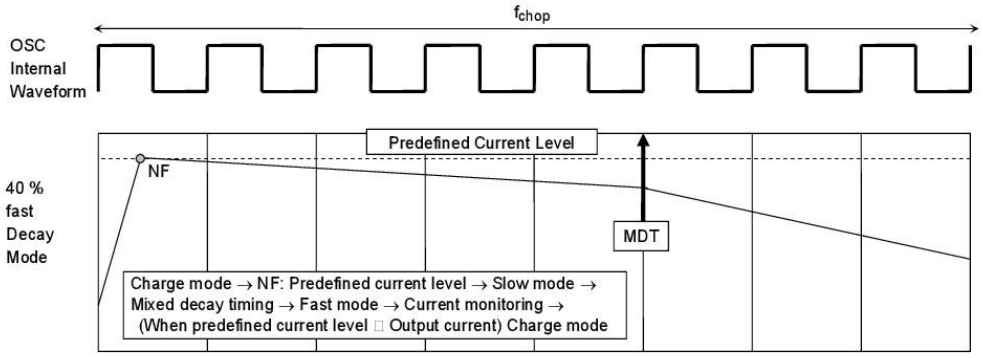


6. Decay Mode

It takes approximately five OSCM cycles for charging-discharging a current in PWM mode. The 40% fast decay mode is created by inducing decay during the last two cycles in Fast Decay mode.
The ratio 40% of the fast decay mode is always fixed.
OSCM = 20 dividing frequency of the master clock (4 MHz, typ.).

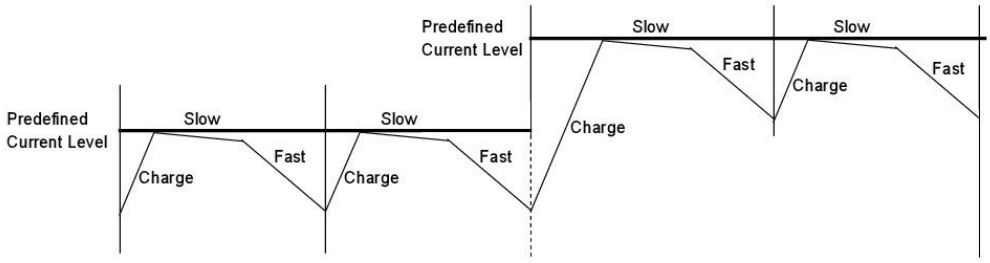
6-1. Current Waveform and Mixed Decay Mode settings

The period of PWM operation is equal to five periods of OSCM.
The ratio 40% of the fast decay mode is always fixed.
The "NF" refers to the point at which the output current reaches its predefined current level.
The smaller the MDT value, the smaller the current ripple amplitude. However, the current decay rate decreases.
MDT means the point of MDT (MIXED DECAY TIMMING) in the below diagram.

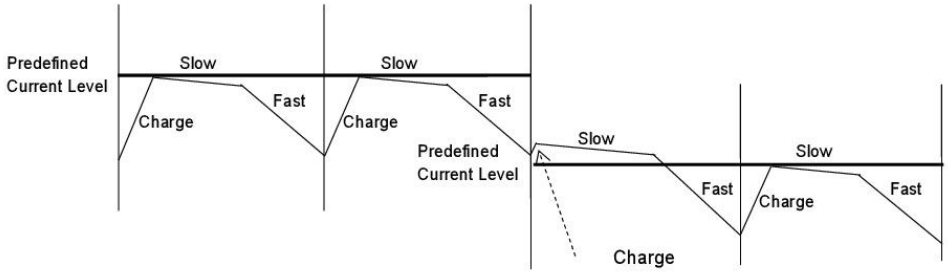


6-2. Effect of Decay Mode

- Increasing the current (sine wave)

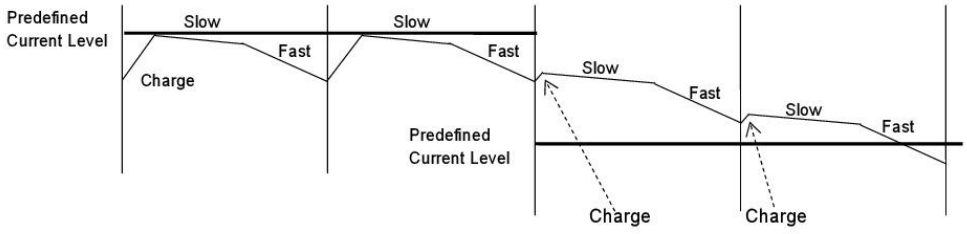


- Decreasing the current (In case the current is decreased to the predefined value in a short time because it decays quickly.)



Even if the output current rises above the predefined current at the RNF point, the current control mode is briefly switched to Charge mode for current sensing.

- Decreasing the current (In case it takes a long time to decrease the current to the predefined value because the current decays slowly.)

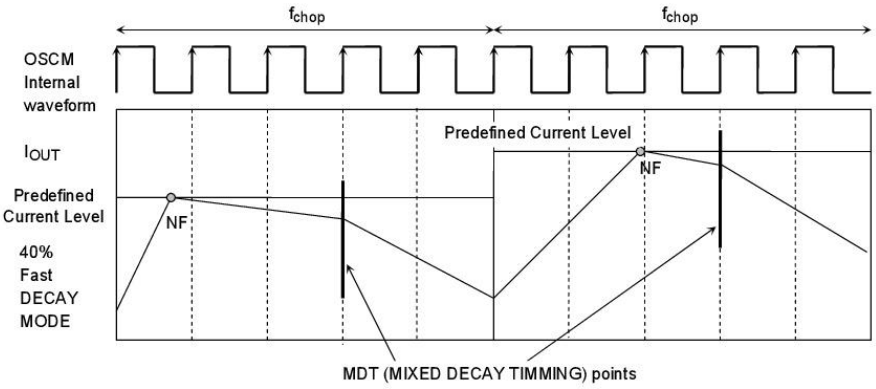


Even if the output current rises above the predefined current at the RNF point, the current control mode is briefly switched to Charge mode for current sensing.

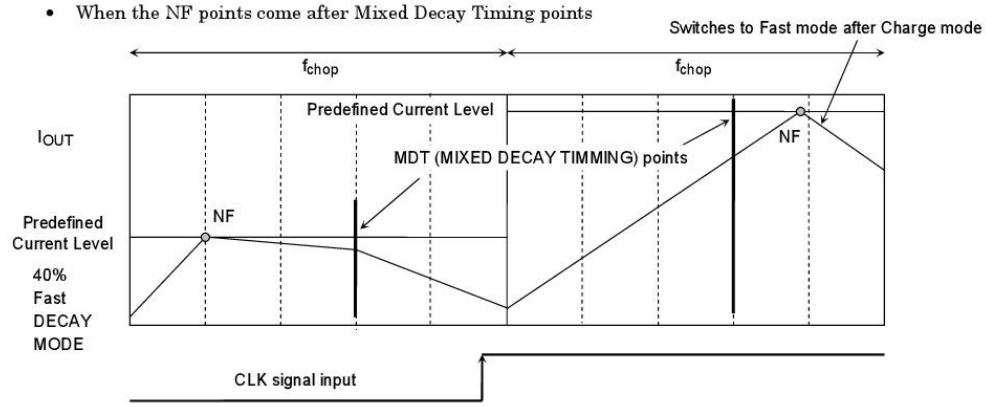
During Mixed Decay and Fast Decay modes, if the predefined current level is less than the output current at the RNF (current monitoring point), the Charge mode in the next chopping cycle will disappear (though the current control mode is briefly switched to Charge mode in actual operations for current sensing) and the current is controlled in Slow and Fast Decay modes (mode switching from Slow Decay mode to Fast Decay mode at the MDT point).

Note: The above figures are rough illustration of the output current. In actual current waveforms, transient response curves can be observed.

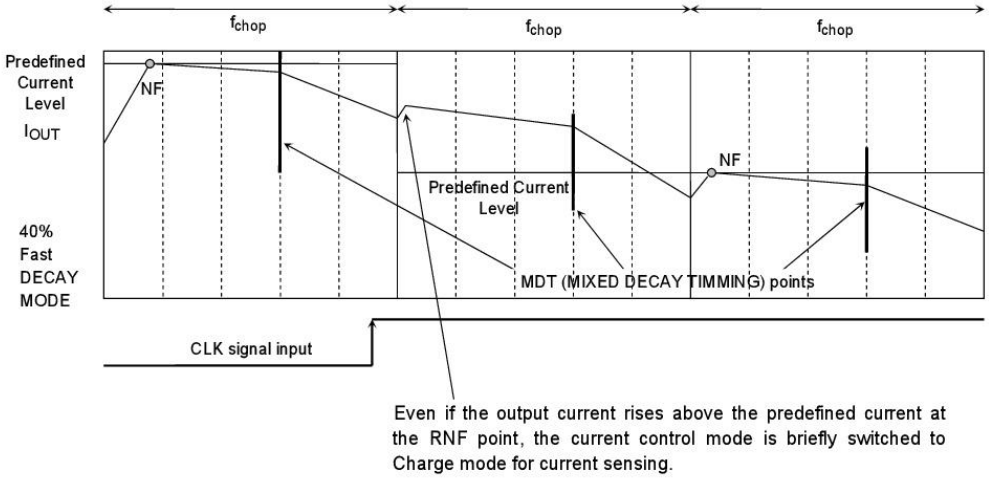
6-3. Current Waveforms in Mixed Decay Mode



- When the NF points come after Mixed Decay Timing points



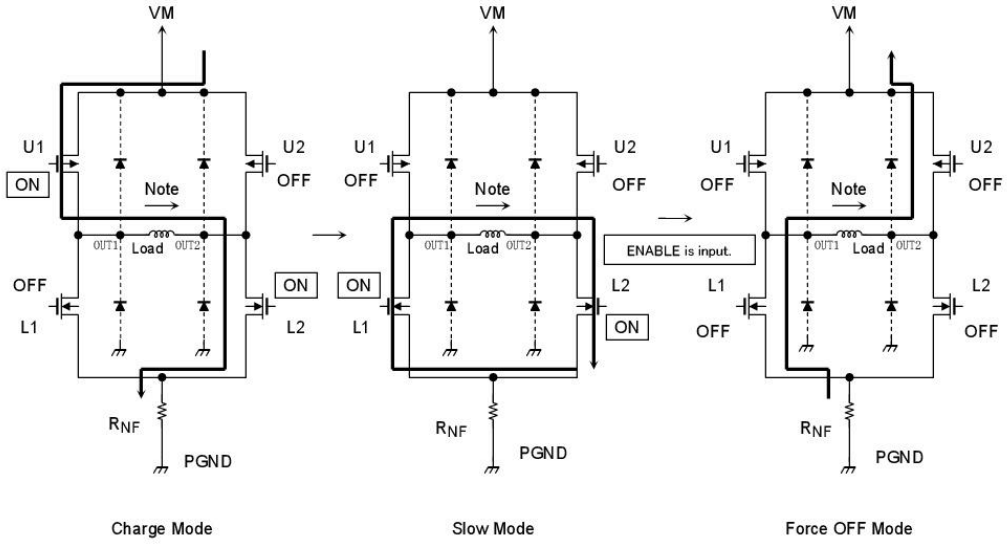
- When the output current value > predefined current level in Mixed Decay mode



Current Draw-out Path when ENABLE is Input in Mid Operation

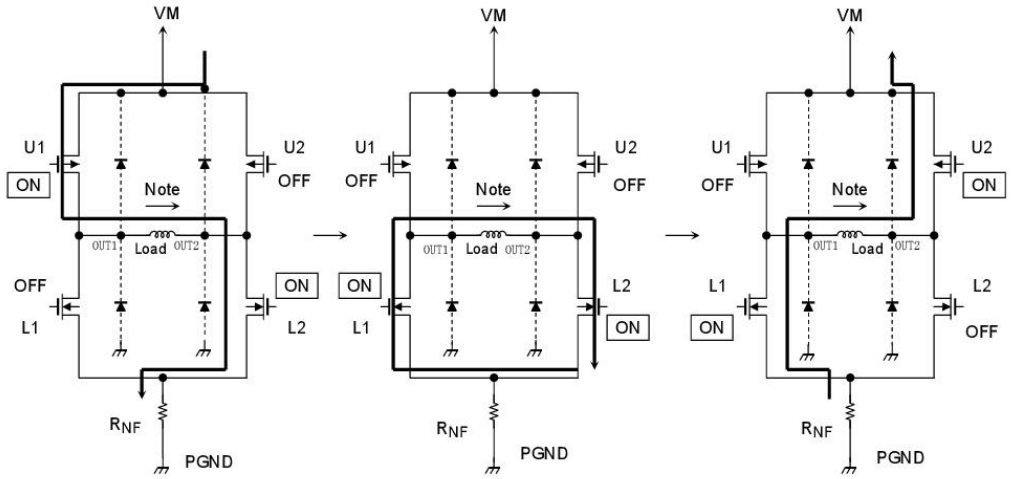
When all the output transistors are forced OFF during Slow mode, the coil energy is drawn out in the following modes:

Note: Parasitic diodes are indicated on the designed lines. However, these are not normally used in Mixed Decay mode.



As shown in the figure above, an output transistor has parasitic diodes.
Normally, when the energy of the coil is drawn out, each transistor is turned ON and the power flows in the opposite-to-normal direction; as a result, the parasitic diode is not used. However, when all the output transistors are forced OFF, the coil energy is drawn out via the parasitic diode.

Output Stage Transistor Operation Mode



Output Stage Transistor Operation Functions

CLK	U1	U2	L1	L2
CHARGE	ON	OFF	OFF	ON
SLOW	OFF	OFF	ON	ON
FAST	OFF	ON	ON	OFF

Note: The above chart shows an example of when the current flows as indicated by the arrows in the above figures. If the current flows in the opposite direction, refer to the following chart:

CLK	U1	U2	L1	L2
CHARGE	OFF	ON	ON	OFF
SLOW	OFF	OFF	ON	ON
FAST	ON	OFF	OFF	ON

Upon transitions of above-mentioned functions, a dead time of about 300 ns (Design target value) is inserted respectively.

Measurement Waveform

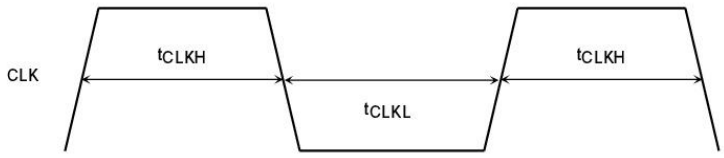


Figure 1 Timing Waveforms and Names

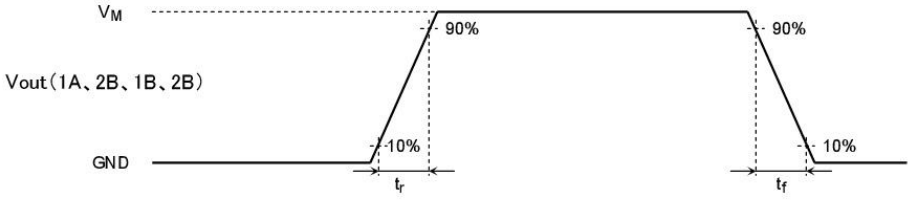


Figure 2 Timing Waveforms and Names

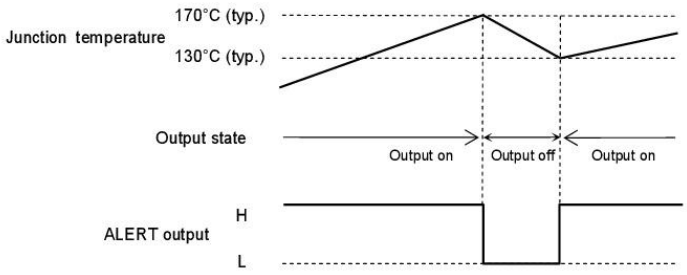
Latch/Auto

Input pin for determining the return method of TSD and ISD.
If Latch/Auto pin outputs low, TSD and ISD functions return by either of turning on power supply again or programming the ENABLE as H → L → H.
If Latch/Auto pin outputs high, they return automatically.
In standby mode, TSD function returns automatically and ISD function cannot operate regardless of the state of the Latch/Auto pin.
When power supply voltage $V_{CC}A$ and $V_{CC}B$ are less than 8 V, TSD and ISD functions cannot operate regardless of the state of the Latch/Auto pin.

Thermal Shut-Down circuit (TSD)

(1) Automatic return

TSD = 170°C (typ.) (Note)
ΔTSD = 40°C (typ.) (Note)

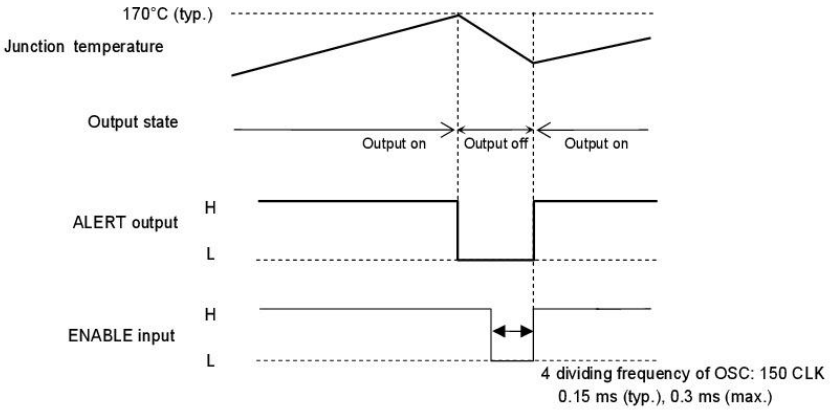


Automatic return has a temperature hysteresis shown in the above figure.

In case of automatic return, the return timing is adjusted at charge start of fshop after the temperature falls to the return temperature (It is 130℃(typ.) in the above figure).
It returns after time passes between 1st and 2nd frequency (fchop).

(2) Latch return

TSD = 170°C (typ.) ± 10 °C (Note)



The operation returns by programming the ENABLE as H → L → H shown in above figure or turning on power supply and turning on UVLO function.

Note: Pre-shipment testing is not performed.

・State of internal IC when TSD circuit operates.

States of internal IC and output correspond to the state in ENABLE mode.
After a return, the timing of output is not determined. It is the same as the case that ENABLE mode is reset.
Operation can start from initial mode by setting the reset low level.

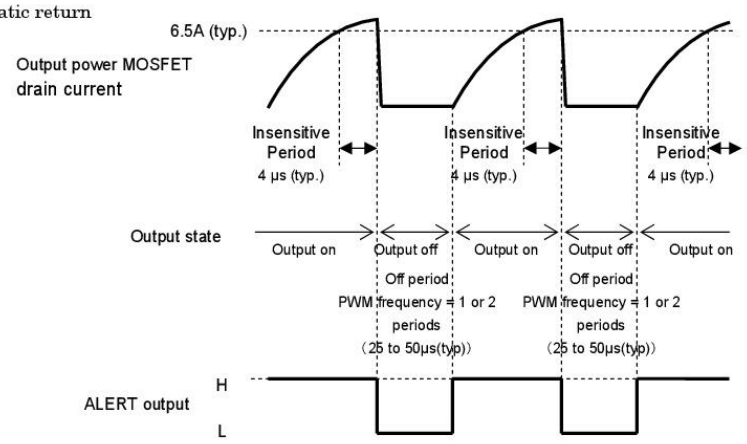
ISD (Over current detection)

Current that flow through output power MOSFETs are monitored individually. If over-current is detected in at least one of all output power MOSFETs, all output power MOSFETs are turned off then this status is kept until ENABLE signal is input. Target value in design is 6.5 A. Pulse of 0.15 ms or more should be recognized.

Masking term of 4μs (typ.) should be provided in order to protect detection error by noise.

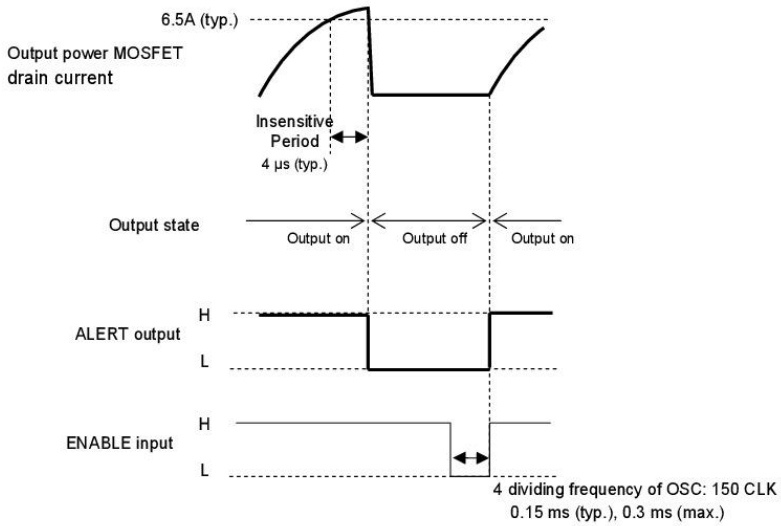
ISD=6.5 A ± 0.15 A (Note)

(1) Automatic return



Operation returns automatically after off period shown in above figure.

(1) Latch return



The operation returns by programming the ENABLE as H → L → H shown in above figure or turning on power supply and turning on UVLO function.

Note: Pre-shipment testing is not performed.

・State of internal IC when ISD circuit operates.

States of internal IC and output correspond to the state in ENABLE mode.
After a return, the timing of output is not determined. It is the same as the case that ENABLE mode is reset.
Operation can start from initial mode by setting the reset low level.

Under Voltage Lock Out (UVLO) circuit

Outputs are shutoff by operating at 5.5 V (Typ.) of V_{CC} or less.
It has a hysteresis of 0.5 V (Typ.) and returns to output when V_{CC} reaches 6.0 V (Typ.).

・State of internal IC when UVLO circuit operates.

The states of the internal IC and outputs correspond to the state in the ENABLE mode and the initial mode at the same time.
After a return, it can start from the initial mode.

When either of $V_{CC}A$ or $V_{CC}B$ falls to around 5.5 V and UVLO operates, output turns off.
It recovers automatically from the initial mode when both V_{CC} rise to around 6.0 V or more.

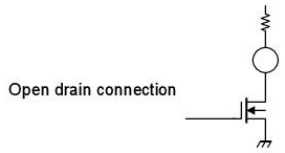
ALERT output

ALERT terminal outputs in detecting TSD or ISD.

ALERT terminal is connected to power supply externally via pull-up resistance.

V_{ALERT} = 0.5 V (max.) at 1 mA

TSD	ISD	ALERT
Under TSD detection	Under TSD detection	Low
Normal	Under TSD detection	
Under TSD detection	Normal	
Normal	Normal	Z



Applied voltage to pull-up resistance is up to 5.5 V. And conducted current is up to 1 mA.

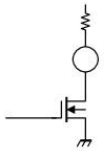
MO output

MO turns on at the predetermined state and output low.

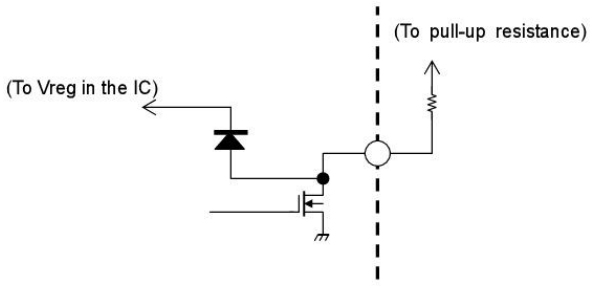
MO terminal is connected to power supply externally via pull-up resistance.

V_{MO} = 0.5 V (max.) at 1 mA

State	MO
Initial	Low
Not initial	Z



Applied voltage to pull-up resistance is up to 5.5 V. And conducted current is up to 1 mA.
It is recommended to gain 5 V by connecting the external pull-up resistance to Vreg pin.



Voltage pull-up of MO and ALERT pins

・It is recommended to pull-up voltage to Vreg pin.

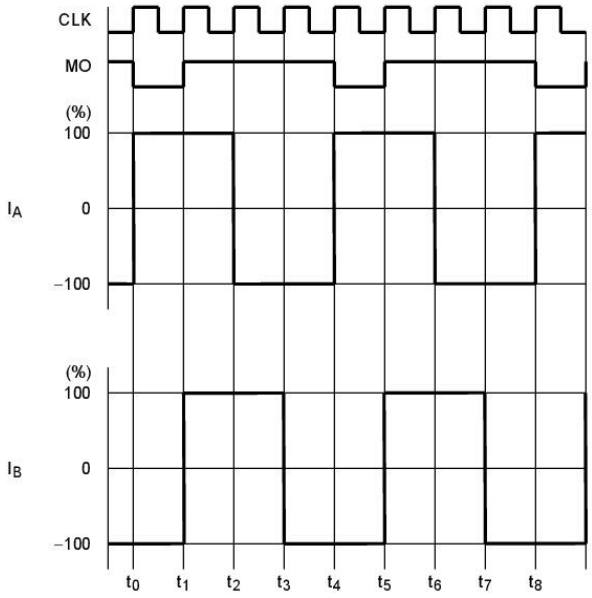
・In case of pull-up to except 5 V (for instance, 3.3 V etc.), it is recommended to use other power supply (ex. 3.3 V) while $V_{CC}A$ and $V_{CC}B$ output between the operation range. When $V_{CC}A$ and $V_{CC}B$ decrease lower than the operation range and Vreg decreases from 5 V to 0 V under the condition that other power supply is used to pull-up voltage, the current continues to conduct from other power supply to the IC inside through the diode shown in the figure. Though this phenomenon does not cause destruction and malfunction of the IC, please consider the set design not to continue such a state for a long time.

・As for the pull-up resistance for MO and ALERT pins, please select large resistance enough for the conducting current so as not to exceed the standard value of 1 mA.

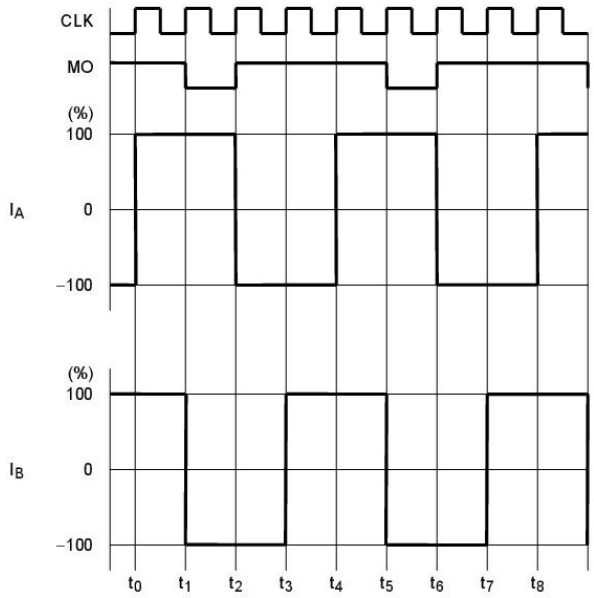
Please use the resistance of 30 kΩ or more in case of applying 5 V, and 20 kΩ or more in case of applying 3.3 V.

Sequence in each excitation mode

1/1-step Excitation Mode (M1: L, M2: L, M3: H, CW Mode)

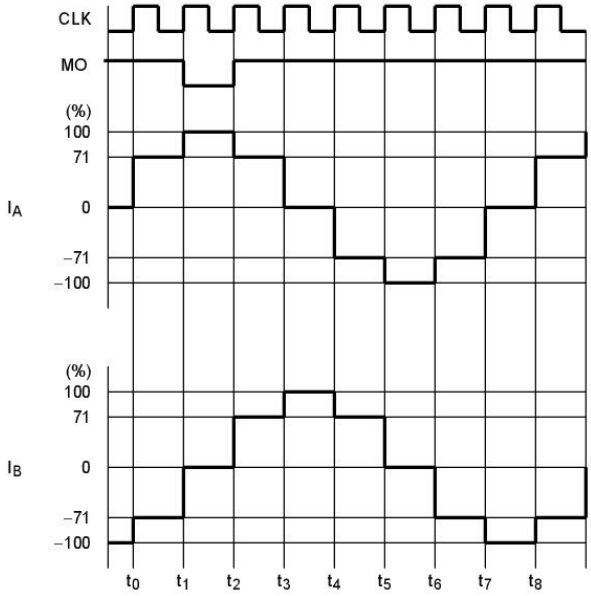


1/1-step Excitation Mode (M1: L, M2: L, M3: H, CCW Mode)

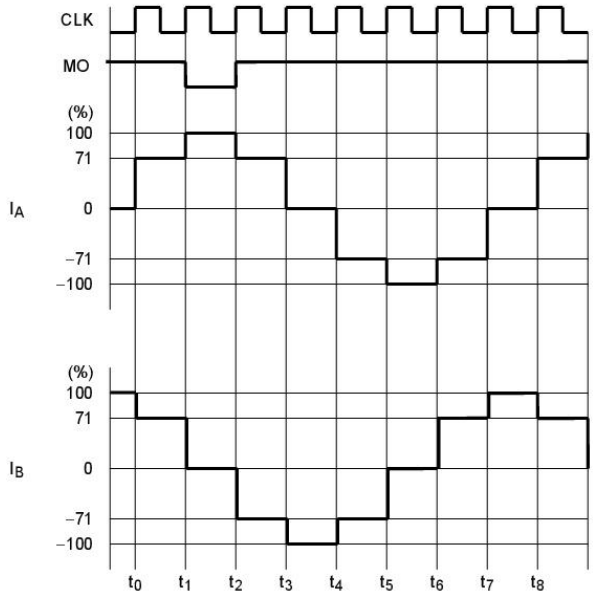


It operates from the initial state after the excitation mode is switched.

1/2-step Excitation Mode (A type) (M1: L, M2: H, M3: L, CW Mode)

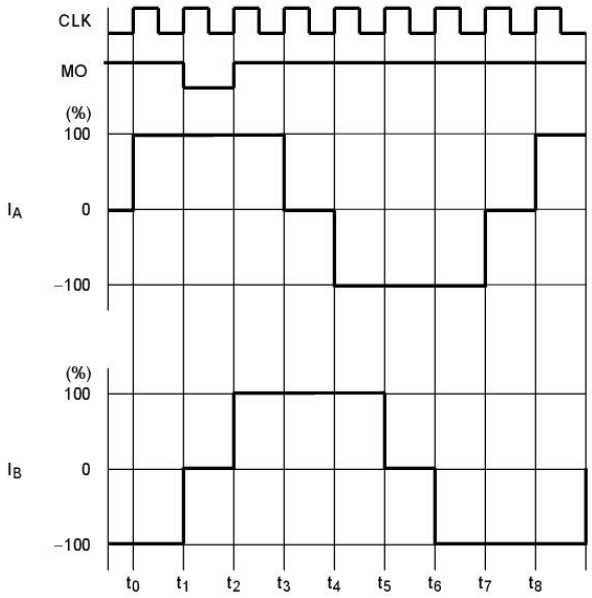


1/2-step Excitation Mode (A type) (M1: L, M2: H, M3: L, CCW Mode)

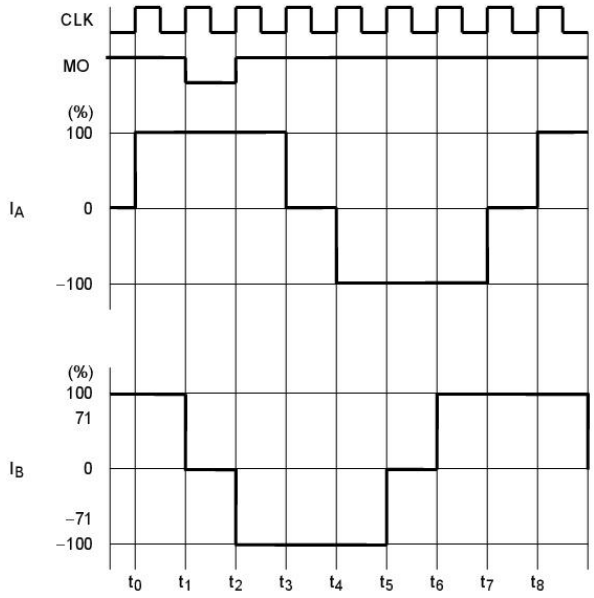


It operates from the initial state after the excitation mode is switched.

1/2-step Excitation Mode (B type) (M1: L, M2: H, M3: H, CW Mode)

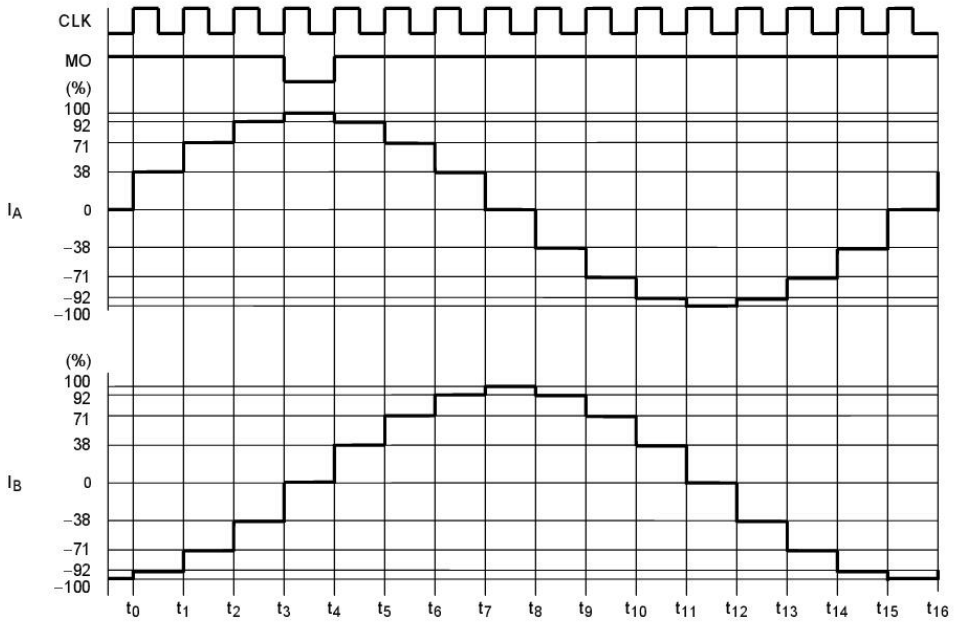


1/2-step Excitation Mode (B type) (M1: L, M2: H, M3: H, CCW Mode)

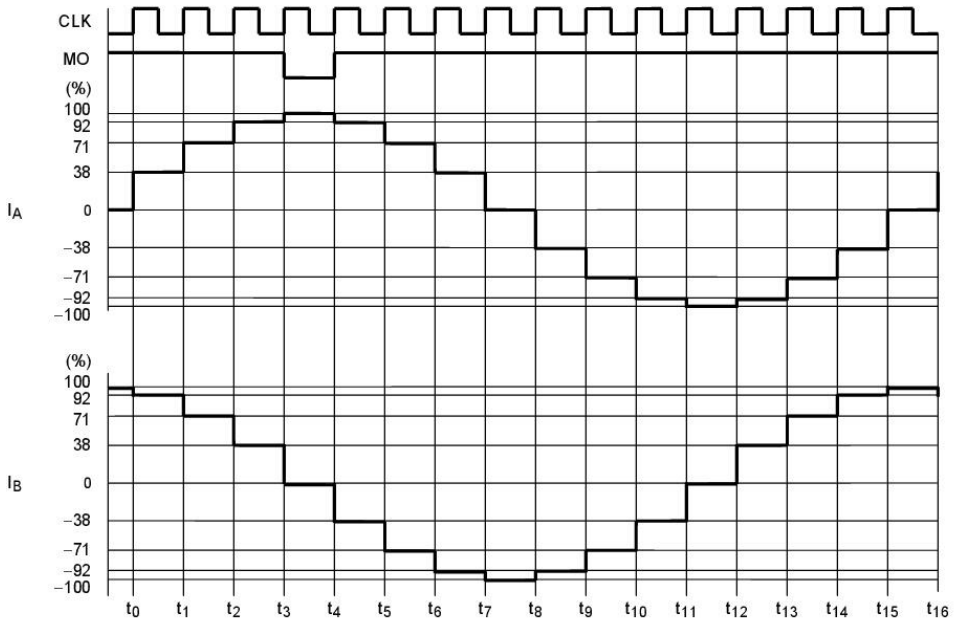


It operates from the initial state after the excitation mode is switched.

1/4-step Excitation Mode (M1: H, M2: L, M3: L, CW Mode)

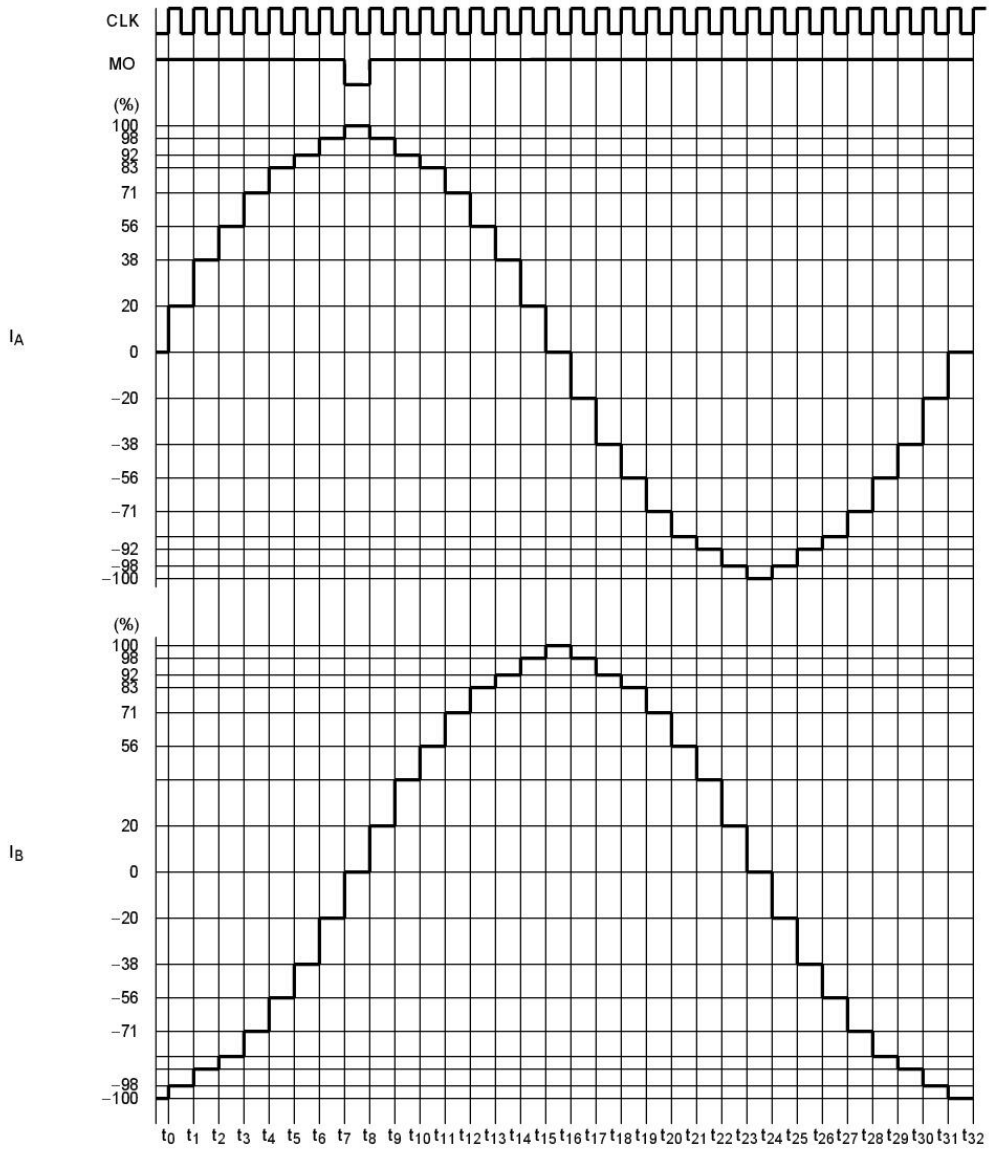


1/4-step Excitation Mode (M1: H, M2: L, M3: L, CCW Mode)



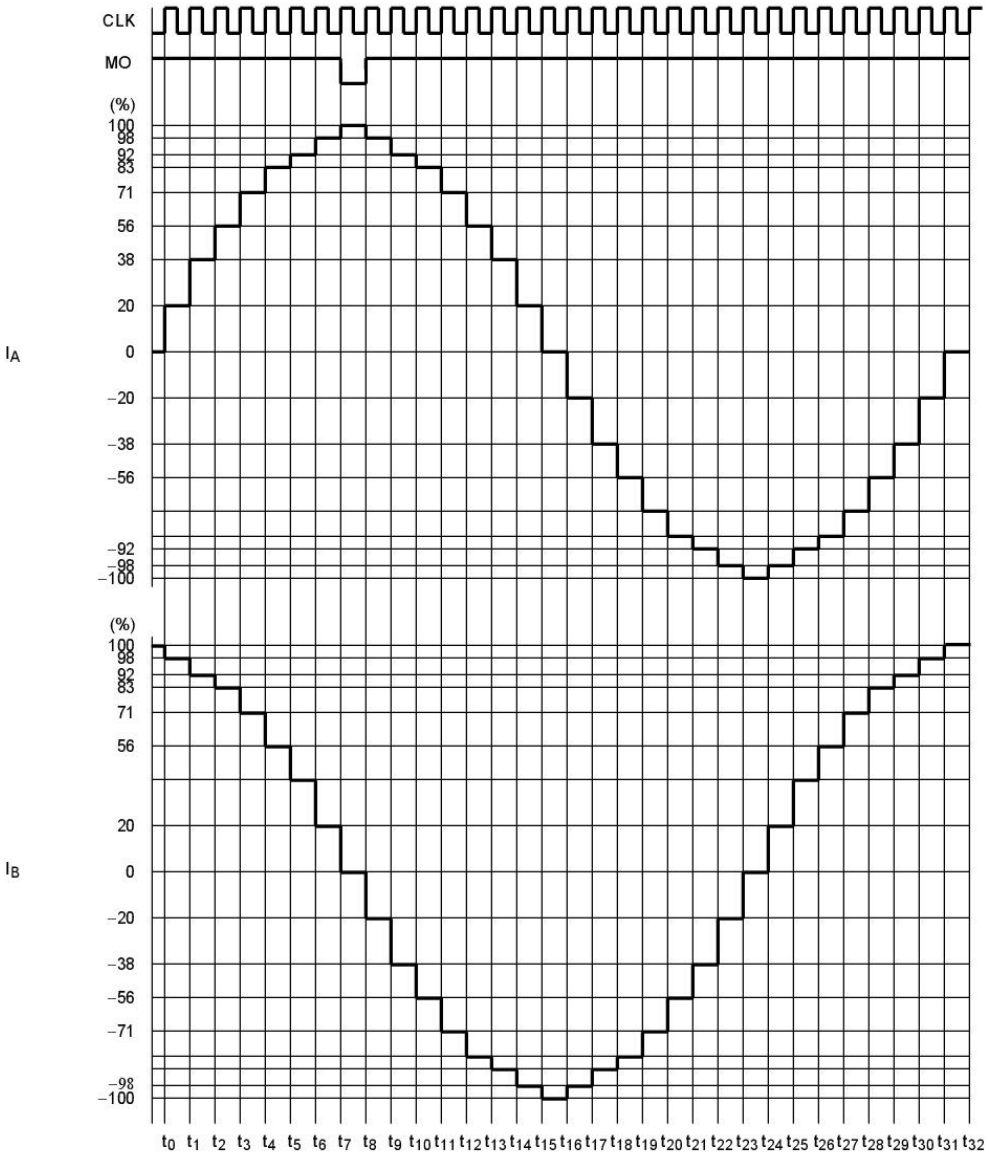
It operates from the initial state after the excitation mode is switched.

1/8-Step Excitation Mode (M1: H, M2: L, M3: H, CW Mode)



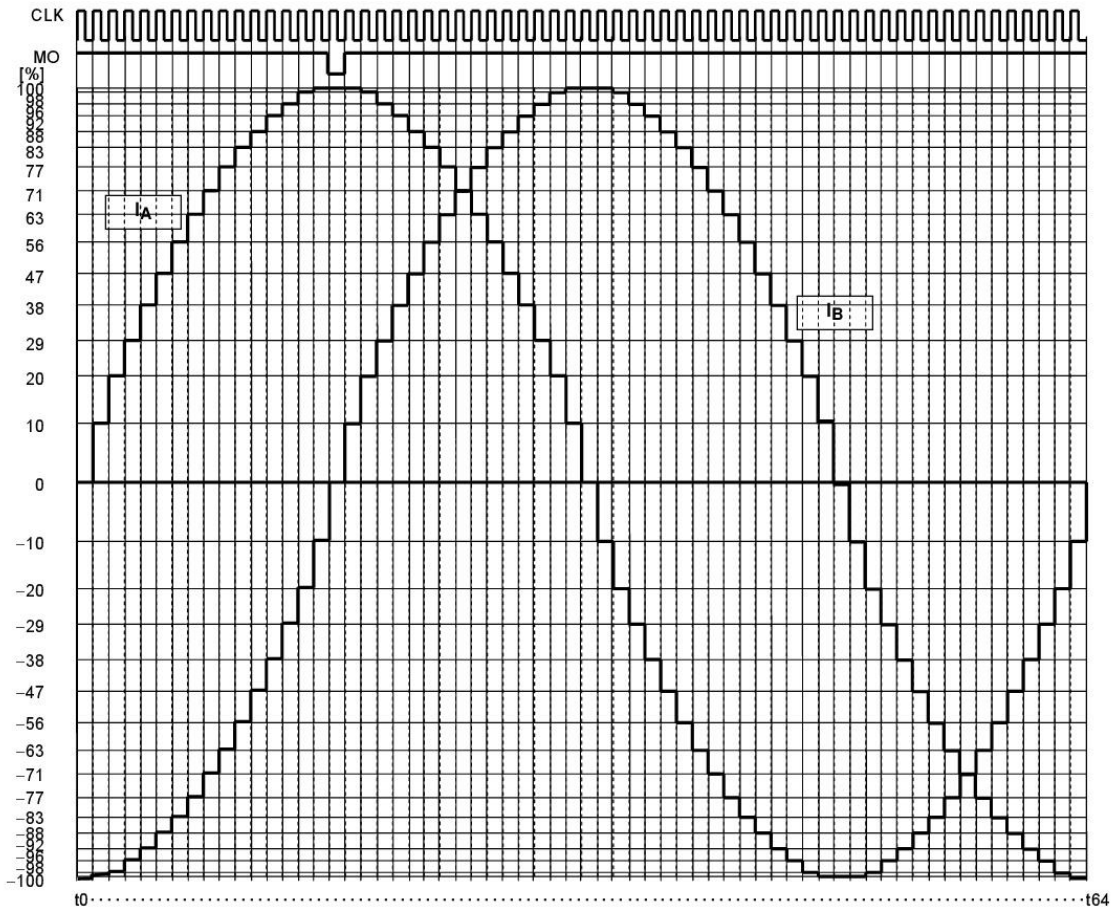
It operates from the initial state after the excitation mode is switched.

1/8-Step Excitation Mode (M1: H, M2: L, M3: H, CCW Mode)



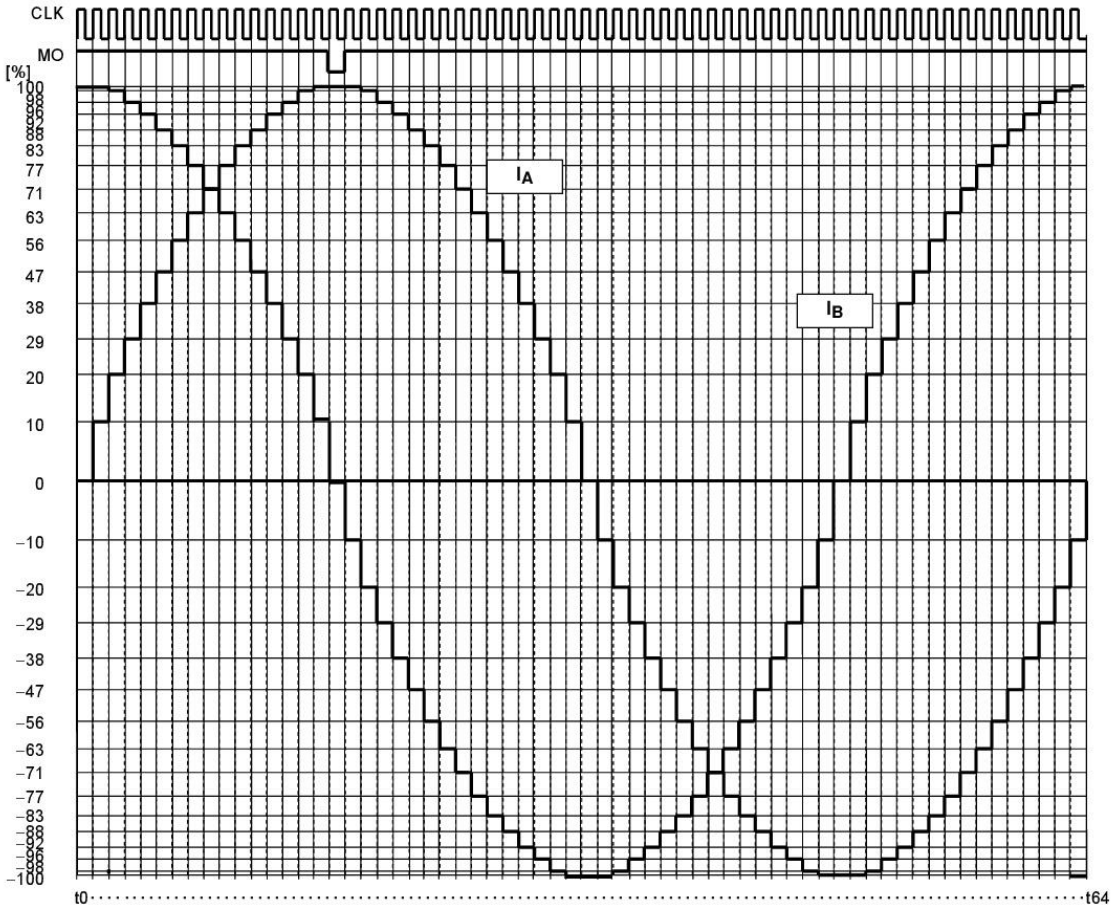
It operates from the initial state after the excitation mode is switched.

1/16-step Excitation Mode (M1: H, M2: H, M3: L, CW Mode)



It operates from the initial state after the excitation mode is switched.

1/16-step Excitation Mode (M1: H, M2: H, M3: L, CCW Mode)



It operates from the initial state after the excitation mode is switched.

Current level

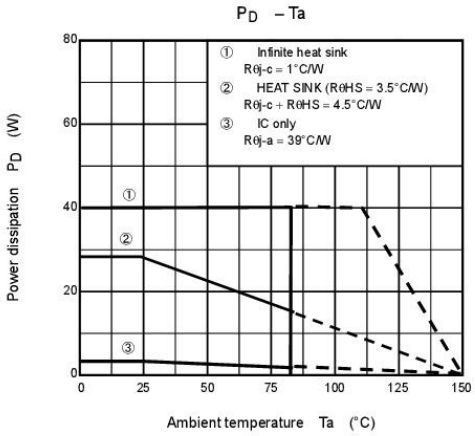
2-phase, 1-2-phase, W1-2-phase, 2W1-2-phase, 4W1-2-phase excitation (unit : %)

Current level (1/16, 1/8, 1/4, 1/2, 1/1)

1/16, 1/8, 1/4, 1/2, 1/1	Min.	Typ.	Max.	Unit
θ 16	---	100.0	---	%
θ 15	95.5	99.5	100.0	
θ 14	94.1	98.1	100.0	
θ 13	91.7	95.7	99.7	
θ 12	88.4	92.4	96.4	
θ 11	84.2	88.2	92.2	
θ 10	79.1	83.1	87.1	
θ 9	73.3	77.3	81.3	
θ 8	66.7	70.7	74.7	
θ 7	59.4	63.4	67.4	
θ 6	51.6	55.6	59.6	
θ 5	43.1	47.1	51.1	
θ 4	34.3	38.3	42.3	
θ 3	25.0	29.0	33.0	
θ 2	15.5	19.5	23.5	
θ 1	5.8	9.8	13.8	
θ 0	---	0.0	---	

Power Dissipation

TB6600HQ

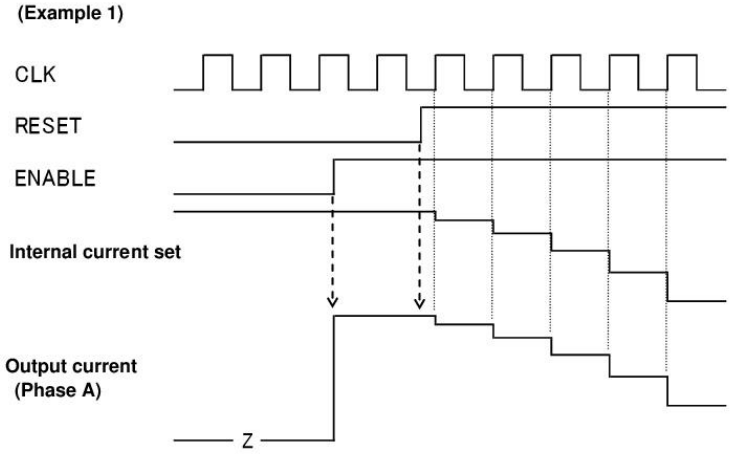


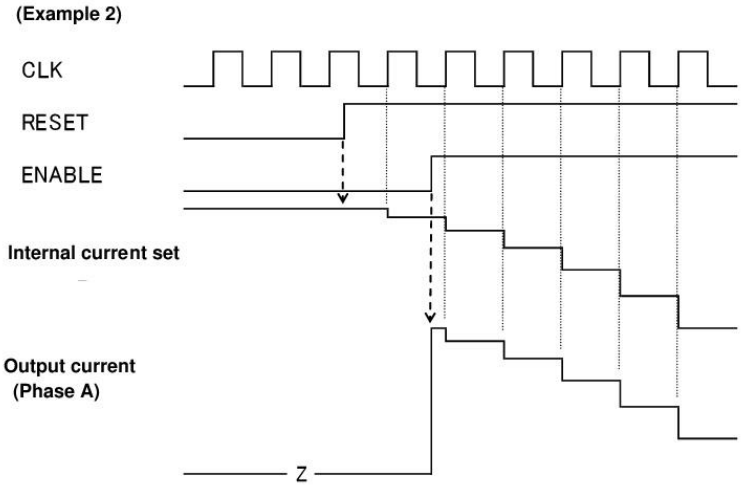
1. How to Turn on the Power

In turning on and off the power, abnormal output is not recognized. Supply monitoring circuit is added in necessary. IC is not damaged in case the order of turning on the power method is not correct. Vref is within the operation range when each input terminal (M1, M2, M3, CLK, CW/CCW, ENABLE, RESET, DCY, TQ, and Latch/Auto) is high or low level and IC is not damaged in turning on the power.
The following is an example. The sequence of turning on the power is not restricted and IC is not damaged when each input pin outputs high and the power supply pin outputs 0 V.
(Example 1): ENABLE = High → RESET = High
(Example 2): RESET = High → ENABLE = High
In example 1, motor can start driving from the initial mode.

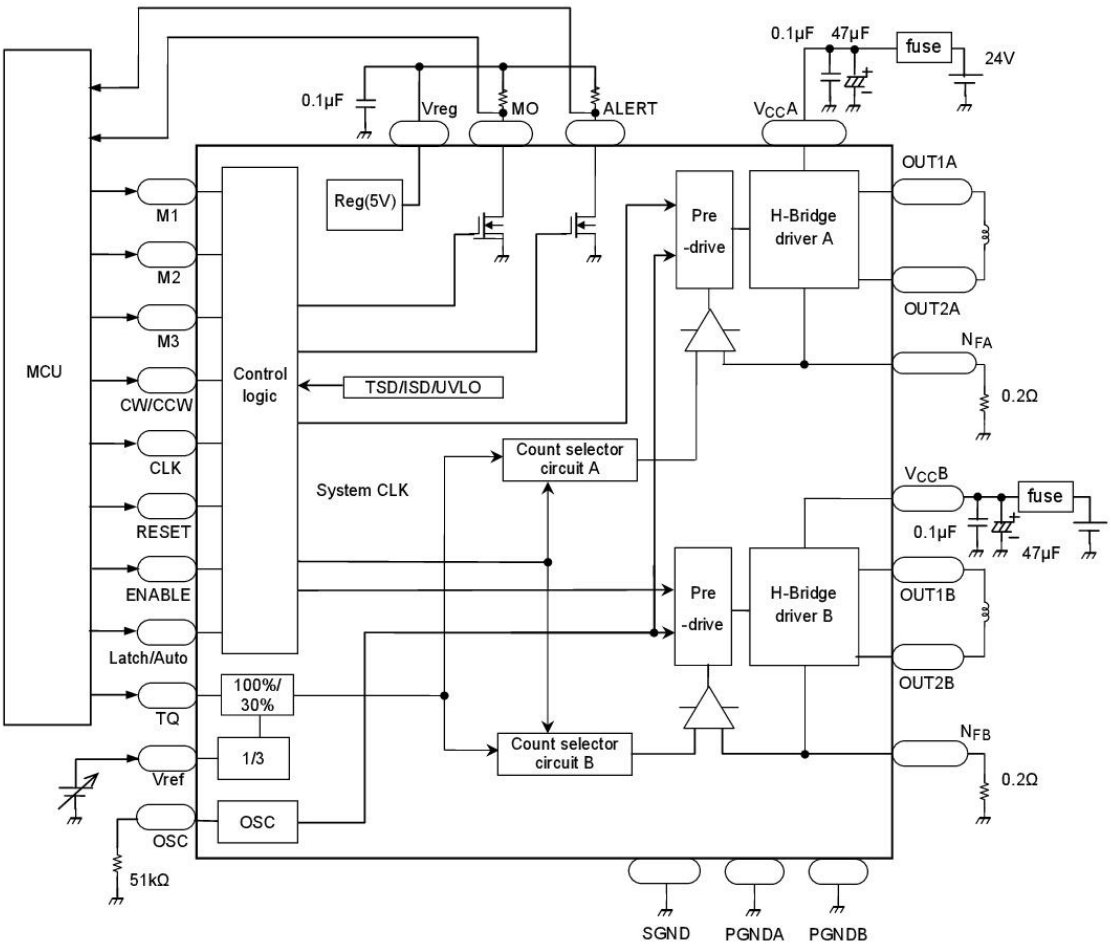
(1) CLK: Current step proceeds to the next mode with respect to every rising edge of CLK.
(2) ENABLE: It is in Hi-Z state in low level. It is output in high level.
RESET: It is in the initial mode (Phase A100% and Phase B %) in low level.
①ENABLE=Low and RESET=Low: Hi-Z. Internal current setting is in initial mode.
②ENABLE=Low and RESET=High: Hi-Z. Internal current setting proceeds by internal counter.
③ENABLE=High and RESET=Low: Output in the initial mode (Phase A100% and Phase B%).
②ENABLE=High and RESET=High: Output at the value which is determined by the internal counter.

<Recommended control input sequence>





Application Circuit

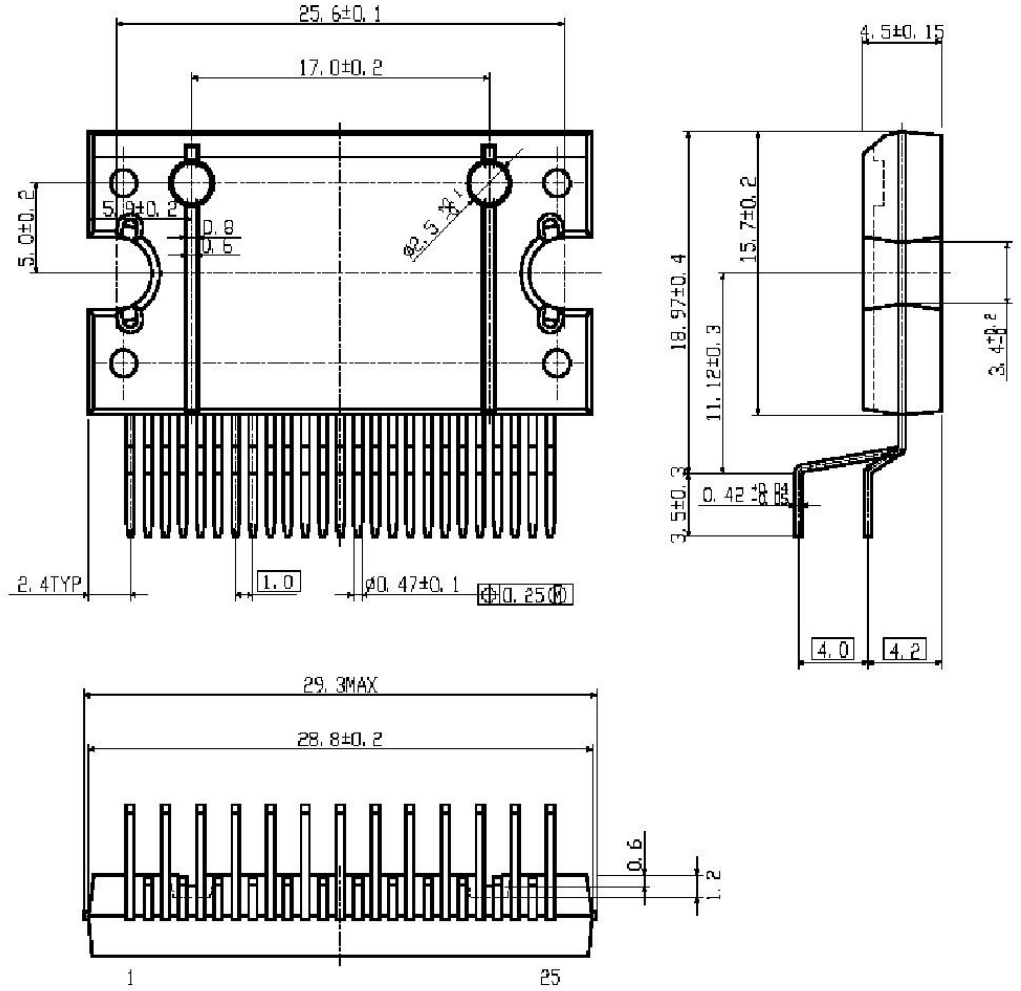


Note 1: Capacitors for the power supply lines should be connected as close to the IC as possible.

Note 2: Current detecting resistances (R_{NFA} and R_{NFB}) should be connected as close to the IC as possible.
External capacitor connected to Vreg: 0.1μF

Package Dimensions

Unit: mm



Weight: 7.7 g (typ.)

Notes on Contents

1. Block Diagrams

Some of the functional blocks, circuits, or constants in the block diagram may be omitted or simplified for explanatory purposes.

2. Equivalent Circuits

The equivalent circuit diagrams may be simplified or some parts of them may be omitted for explanatory purposes.

3. Timing Charts

Timing charts may be simplified for explanatory purposes.

4. Application Circuits

The application circuits shown in this document are provided for reference purposes only. Thorough evaluation is required, especially at the mass production design stage.
Toshiba does not grant any license to any industrial property rights by providing these examples of application circuits.

5. Test Circuits

Components in the test circuits are used only to obtain and confirm the device characteristics. These components and circuits are not guaranteed to prevent malfunction or failure from occurring in the application equipment.

IC Usage Considerations

Notes on handling of ICs

[1] The absolute maximum ratings of a semiconductor device are a set of ratings that must not be exceeded, even for a moment. Do not exceed any of these ratings.
Exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.

[2] Use an appropriate power supply fuse to ensure that a large current does not continuously flow in case of over current and/or IC failure. The IC will fully break down when used under conditions that exceed its absolute maximum ratings, when the wiring is routed improperly or when an abnormal pulse noise occurs from the wiring or load, causing a large current to continuously flow and the breakdown can lead smoke or ignition. To minimize the effects of the flow of a large current in case of breakdown, appropriate settings, such as fuse capacity, fusing time and insertion circuit location, are required.

[3] If your design includes an inductive load such as a motor coil, incorporate a protection circuit into the design to prevent device malfunction or breakdown caused by the current resulting from the inrush current at power ON or the negative current resulting from the back electromotive force at power OFF. IC breakdown may cause injury, smoke or ignition.
Use a stable power supply with ICs with built-in protection functions. If the power supply is unstable, the protection function may not operate, causing IC breakdown. IC breakdown may cause injury, smoke or ignition.

[4] Do not insert devices in the wrong orientation or incorrectly.
Make sure that the positive and negative terminals of power supplies are connected properly.
Otherwise, the current or power consumption may exceed the absolute maximum rating, and exceeding the rating(s) may cause the device breakdown, damage or deterioration, and may result injury by explosion or combustion.
In addition, do not use any device that is applied the current with inserting in the wrong orientation or incorrectly even just one time.

Points to remember on handling of ICs

(1) Over current Protection Circuit

Over current protection circuits (referred to as current limiter circuits) do not necessarily protect ICs under all circumstances. If the over current protection circuits operate against the over current, clear the over current status immediately.

Depending on the method of use and usage conditions, such as exceeding absolute maximum ratings can cause the over current protection circuit to not operate properly or IC breakdown before operation. In addition, depending on the method of use and usage conditions, if over current continues to flow for a long time after operation, the IC may generate heat resulting in breakdown.

(2) Thermal Shutdown Circuit

Thermal shutdown circuits do not necessarily protect ICs under all circumstances. If the thermal shutdown circuits operate against the over temperature, clear the heat generation status immediately.

Depending on the method of use and usage conditions, such as exceeding absolute maximum ratings can cause the thermal shutdown circuit to not operate properly or IC breakdown before operation.

(3) Heat Radiation Design

In using an IC with large current flow such as power amp, regulator or driver, please design the device so that heat is appropriately radiated, not to exceed the specified junction temperature (T_j) at any time and condition. These ICs generate heat even during normal use. An inadequate IC heat radiation design can lead to decrease in IC life, deterioration of IC characteristics or IC breakdown. In addition, please design the device taking into considerate the effect of IC heat radiation with peripheral components.

(4) Back-EMF

When a motor rotates in the reverse direction, stops or slows down abruptly, a current flow back to the motor's power supply due to the effect of back-EMF. If the current sink capability of the power supply is small, the device's motor power supply and output pins might be exposed to conditions beyond maximum ratings. To avoid this problem, take the effect of back-EMF into consideration in system design.

(5) Others

Utmost care is necessary in the design of the output, V_{CC}, VM, and GND lines since the IC may be destroyed by short-circuiting between outputs, air contamination faults, or faults due to improper grounding, or by short-circuiting between contiguous pins.

9.2.3 Sensor de HOME

Slotted Interrupter
Gabellichtschranke
Version 1.0

SFH 9500

Features:
- Suitable for surface mounting (SMT)
- Compact housing out of black LCP
- GaAs infrared emitter (950 nm)
- Silicon phototransistor with daylight-cutoff filter
- With positioning pin
- Suitable for pick and place
- High sensing accuracy (slit width: 0.5 mm)
- Wide gap between emitter and detector (5 mm)
- High stability on pcb due to large width of device (6.8 mm)

Besondere Merkmale:
- Geeignet für Oberflächenmontage (SMT)
- Kompaktes Gehäuse aus schwarzem LCP
- GaAs-IR-Sendediode (950 nm)
- Si-Fototransistor mit Tageslichtsperrfilter
- Mit Positionspin
- Geeignet für "pick and place" Montage
- Hohe Genauigkeit (Schlitzbreite: 0,5 mm)
- Große Spaltbreite zwischen Sender und Empfänger (5 mm)
- Hohe Stabilität auf PCB durch große Bauelementabmessung (6,8 mm)

Applications
- Speed control
- Motor control
- Monitoring of paper feed in printers, copiers, facsimiles
- Disk drives
- Control of print head in printers
- Coin detection
- Optoelectronic switches

Anwendungen
- Geschwindigkeitsüberwachung
- Motorsteuerung
- Überwachung des Papiervorschubs in Druckern, Kopier- und Faxgeräten
- Speicherlaufwerke
- Steuerung des Druckkopfes in Druckern
- Münzdetektion
- Optoelektronische Schalter

Ordering Information
Bestellinformation

Type: Typ:	Collector-emitter current Kollektor-Emitterstrom $I_F = 20$ mA, $V_{CE} = 5$ V I_{PCE} [µA]	Ordering Code Bestellnummer
SFH 9500	≥ 1000	Q65110A3108

Maximum Ratings (T_A = 25 °C)
Grenzwerte

Parameter Bezeichnung	Symbol Symbol	Values Werte	Unit Einheit
Emitter **Sender**			
Reverse voltage Sperrspannung	V_R	5	V
Forward current Durchlassstrom	I_F	60	mA
Total power dissipation Verlustleistung	P_{tot}	100	mW
Thermal resistance junction - ambient [1) page 10] Wärmewiderstand Sperrschicht - Umgebung 1) *Seite 10*	R_{thJA}	280	K / W
Detector **Empfänger**			
Collector-emitter voltage Kollektor-Emitter-Spannung	V_{CE}	30	V
Collector-emitter voltage Kollektor-Emitter-Spannung (t ≤ 2 min)	V_{CE}	70	V
Emitter-collector voltage Emitter-Kollektor-Spannung	V_{EC}	7	V
Collector current Kollektorstrom	I_C	50	mA
Total power dissipation Verlustleistung	P_{tot}	150	mW
Thermal resistance junction - ambient [1) page 10] Wärmewiderstand Sperrschicht - Umgebung 1) *Seite 10*	R_{thJA}	280	K / W
Slotted Interrupter **Gabellichtschranke**			
Storage temperature range Lagertemperatur	T_{stg}	-40 ... 85	°C

Version 1.0 **SFH 9500**

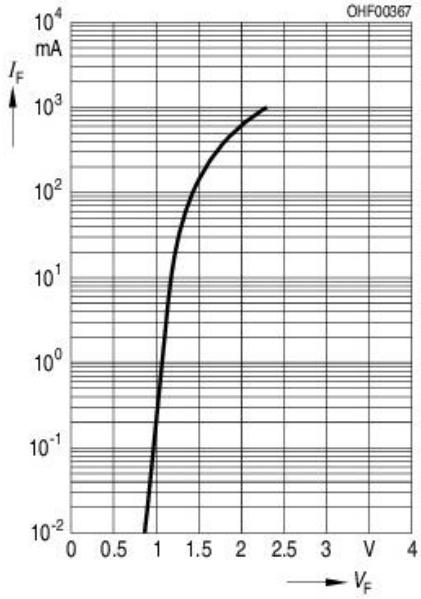
Parameter Bezeichnung	Symbol Symbol	Values Werte	Unit Einheit
Ambient temperature range Umgebungstemperatur	T_{op}	-40 ... 85	°C
Electrostatic discharge Elektrostatische Entladung	V_{ESD}	2	kV
Thermal resistance junction - ambient Wärmewiderstand Sperrschicht - Umgebung	R_{thJA}	280	K / W

Characteristics (T_A = 25 °C)
Kennwerte

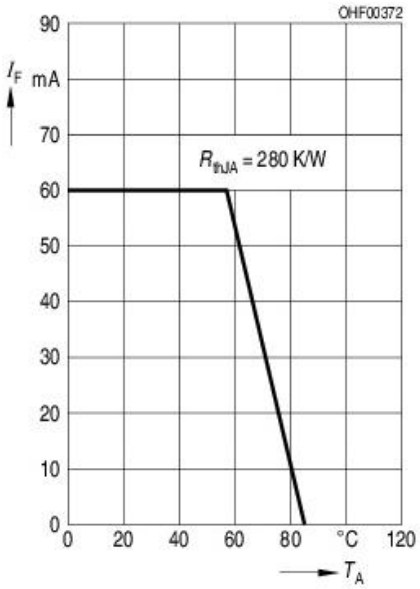
Parameter Bezeichnung	Symbol Symbol	Values Werte	Unit Einheit
Emitter **Sender**			
Emission wavelength Zentrale Emissionswellenlänge	λ_{peak}	950	nm
Forward voltage Durchlassspannung (I_F = 20 mA, t_p = 20 ms)	V_F	1.2 (≤ 1.4)	V
Reverse current Sperrstrom (V_R = 5 V)	I_R	0.01 (≤ 1)	µA
Capacitance Kapazität (V_R = 0 V, f = 1 MHz)	C_0	16	pF
Detector **Empfänger**			
Wavelength of max. sensitivity Wellenlänge der max. Fotoempfindlichkeit	$\lambda_{S\,max}$	920	nm
Spectral range of sensitivity Spektraler Bereich der Fotoempfindlichkeit	$\lambda_{10\%}$	840 ... 1080	nm
Capacitance Kapazität (V_{CE} = 0 V, f = 1 MHz, E = 0)	C_{CE}	6.5	pF

Parameter Bezeichnung	Symbol Symbol	Values Werte	Unit Einheit
Dark current Dunkelstrom (V_{CE} = 20 V)	I_{CE0}	2 (≤ 50)	nA
Interrupter Lichtschranke			
Collector-emitter current Kollektor-Emitterstrom (I_F = 20 mA, V_{CE} = 5 V)	I_{PCE}	1000	µA
Collector-emitter saturation voltage Kollektor-Emitter Sättigungsspannung (I_F = 20 mA, I_C = 0.3 mA)	V_{CEsat}	≤ 0.4	mV
Switching Times Schaltzeiten			
Rise time Anstiegzeit (V_{CC} = 5 V, I_C = 1 mA, R_L = 1 kΩ)	t_r	13	µs
Fall time Abfallzeit (V_{CC} = 5 V, I_C = 1 mA, R_L = 1 kΩ)	t_f	17	µs

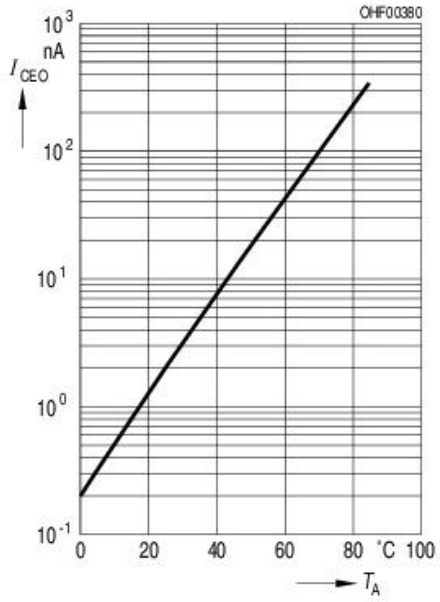
Forward Current
Durchlassstrom
$I_F = f(V_F)$, single pulse, $t_p = 100\ \mu s$, $T_A = 25°C$



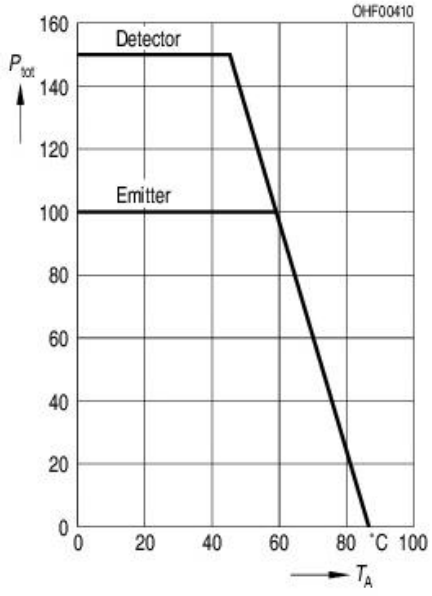
Max. Permissible Forward Current
Max. zulässiger Durchlassstrom
$I_F = f(T_A)$



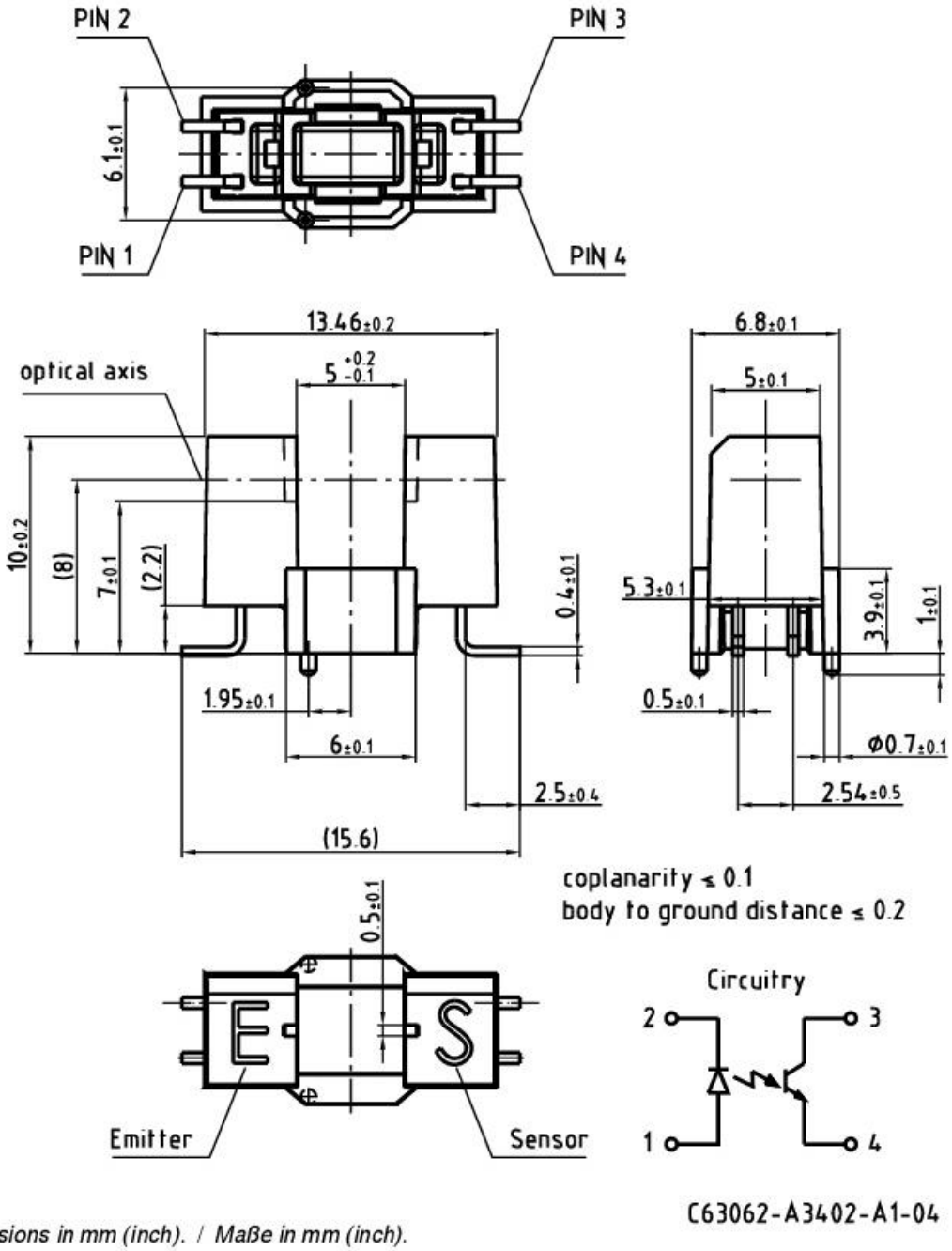
Dark Current
Dunkelstrom
$I_{CEO} = f(T_A)$, $V_{CE} = 20$ V, $E = 0$



Total Power Dissipation
Verlustleistung
$P_{tot} = f(T_A)$

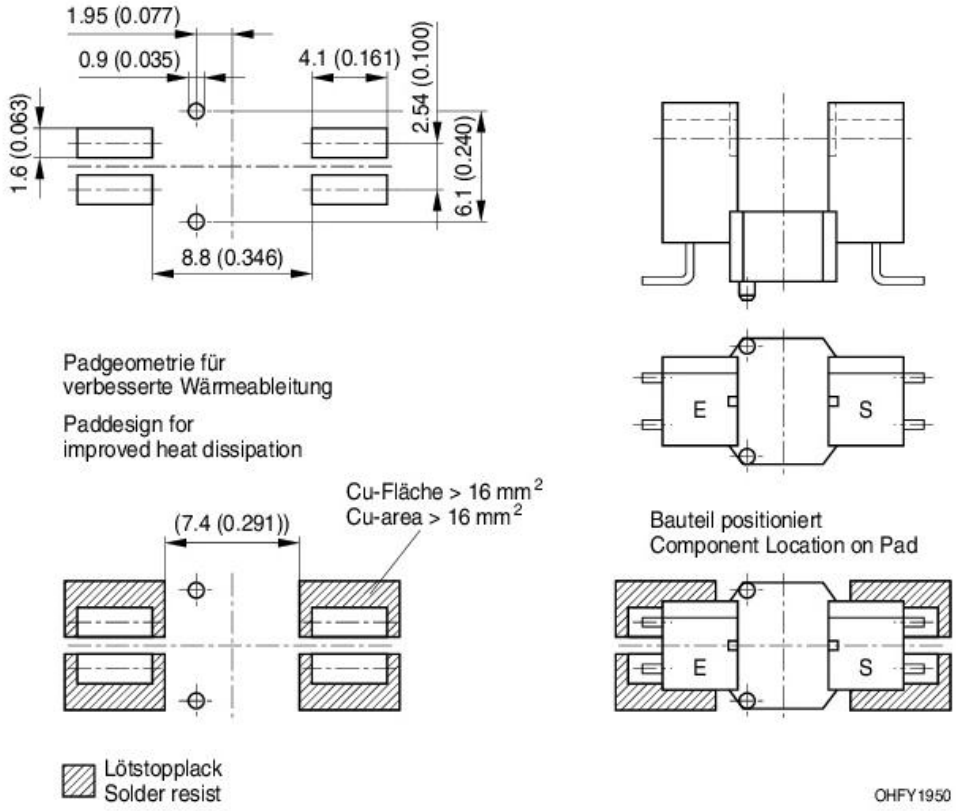


Package Outline
Maßzeichnung

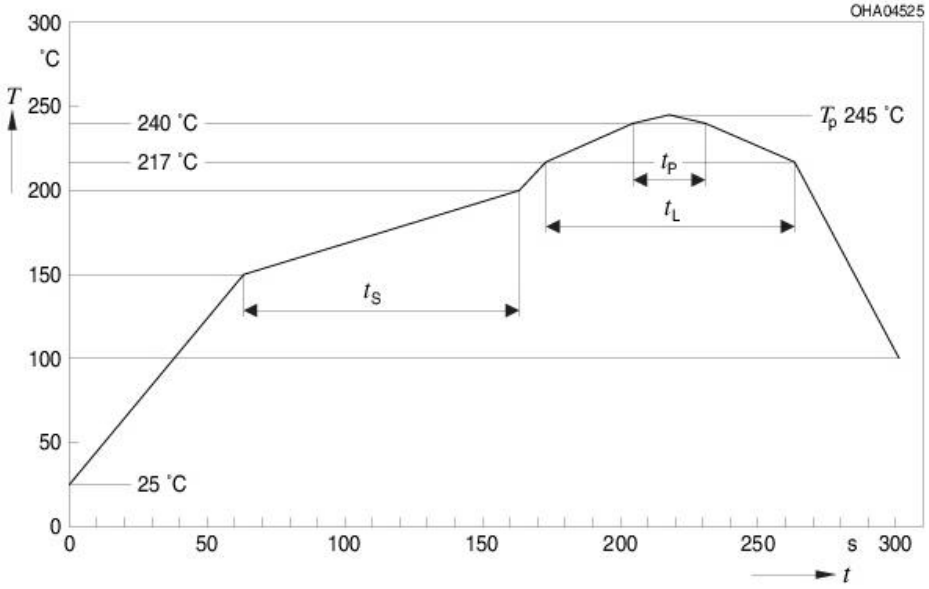


Dimensions in mm (inch). / Maße in mm (inch).

Recommended Solder Pad
Empfohlenes Lötpaddesign
/



Reflow Soldering Profile
Reflow-Lötprofil
Preconditioning: JEDEC Level 1 acc. to JEDEC J-STD-020D.01



OHA04612

Profile Feature Profil-Charakteristik	Symbol Symbol	Pb-Free (SnAgCu) Assembly			Unit Einheit
		Minimum	Recommendation	Maximum	
Ramp-up rate to preheat*) 25 °C to 150 °C			2	3	K/s
Time t_S T_{Smin} to T_{Smax}	t_S	60	100	120	s
Ramp-up rate to peak*) T_{Smax} to T_P			2	3	K/s
Liquidus temperature	T_L		217		°C
Time above liquidus temperature	t_L		80	100	s
Peak temperature	T_P		245	260	°C
Time within 5 °C of the specified peak temperature T_P - 5 K	t_P	10	20	30	s
Ramp-down rate* T_P to 100 °C			3	6	K/s
Time 25 °C to T_P				480	s

All temperatures refer to the center of the package, measured on the top of the component
* slope calculation DT/Dt: Dt max. 5 s; fulfillment for the whole T-range

Disclaimer

OSRAM OS assumes no liability whatsoever for any use of this document or its content by recipient including, but not limited to, for any design in activities based on this preliminary draft version. OSRAM OS may e.g. decide at its sole discretion to stop developing and/or finalising the underlying design at any time.

Attention please!
The information describes the type of component and shall not be considered as assured characteristics.
Terms of delivery and rights to change design reserved.
Due to technical requirements components may contain dangerous substances.
For information on the types in question please contact our Sales Organization.
If printed or downloaded, please find the latest version in the Internet.
Packing
Please use the recycling operators known to you. We can also help you – get in touch with your nearest sales office.
By agreement we will take packing material back, if it is sorted. You must bear the costs of transport. For packing material that is returned to us unsorted or which we are not obliged to accept, we shall have to invoice you for any costs incurred.
Components used in life-support devices or systems must be expressly authorized for such purpose!
Critical components* may only be used in life-support devices** or systems with the express written approval of OSRAM OS.

*) A critical component is a component used in a life-support device or system whose failure can reasonably be expected to cause the failure of that life-support device or system, or to affect its safety or the effectiveness of that device or system.
**) Life support devices or systems are intended (a) to be implanted in the human body, or (b) to support and/or maintain and sustain human life. If they fail, it is reasonable to assume that the health and the life of the user may be endangered.

Disclaimer

OSRAM OS übernimmt keine wie auch immer geartete Haftung für die Nutzung dieses Dokuments und seines Inhaltes durch den Empfänger, insbesondere nicht für irgendwelche Design-Aktivitäten, die auf dieser vorläufigen Entwurfsversion basieren. OSRAM OS behält sich beispielsweise auch vor, jederzeit die Weiter- und Fertigentwicklung des zugrundeliegenden Designs einseitig einzustellen.

Bitte beachten!
Lieferbedingungen und Änderungen im Design vorbehalten. Aufgrund technischer Anforderungen können die Bauteile Gefahrstoffe enthalten. Für weitere Informationen zu gewünschten Bauteilen, wenden Sie sich bitte an unseren Vertrieb. Falls Sie dieses Datenblatt ausgedruckt oder heruntergeladen haben, finden Sie die aktuellste Version im Internet.
Verpackung
Benutzen Sie bitte die Ihnen bekannten Recyclingwege. Wenn diese nicht bekannt sein sollten, wenden Sie sich bitte an das nächstgelegene Vertriebsbüro. Wir nehmen das Verpackungsmaterial zurück, falls dies vereinbart wurde und das Material sortiert ist. Sie tragen die Transportkosten. Für Verpackungsmaterial, das unsortiert an uns zurückgeschickt wird oder das wir nicht annehmen müssen, stellen wir Ihnen die anfallenden Kosten in Rechnung.
Bauteile, die in lebenserhaltenden Apparaten und Systemen eingesetzt werden, müssen für diese Zwecke ausdrücklich zugelassen sein!
Kritische Bauteile* dürfen in lebenserhaltenden Apparaten und Systemen** nur dann eingesetzt werden, wenn ein schriftliches Einverständnis von OSRAM OS vorliegt.

*) Ein kritisches Bauteil ist ein Bauteil, das in lebenserhaltenden Apparaten oder Systemen eingesetzt wird und dessen Defekt voraussichtlich zu einer Fehlfunktion dieses lebenserhaltenden Apparates oder Systems führen wird oder die Sicherheit oder Effektivität dieses Apparates oder Systems beeinträchtigt.
**) Lebenserhaltende Apparate oder Systeme sind für (a) die Implantierung in den menschlichen Körper oder (b) für die Lebenserhaltung bestimmt. Falls Sie versagen, kann davon ausgegangen werden, dass die Gesundheit und das Leben des Patienten in Gefahr ist.

9.2.4 Sensor de Barrera

M18 SERIES
Cost-Effective Cylindrical Photoelectrical Sensors

Panasonic
ideas for life

Cylindrical Type – Complete Range of M18 Sensors

CE Marked
Conforming to EMC Directive

Cost-Effective

Thanks to a fully automated production line, M18 Series Sensors offer an excellent quality at low cost. With their widely-used M18 cylindrical shape, they grant for a quick and easy installation.

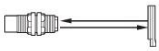
Complete Range

Metal or plastic housing
- PNP or NPN output
- Connector or cable type
- Thru beam, retroreflective with or without polarising filter and diffuse reflective type available
- Various cables, reflectors and holders

Sensitivity Adjuster

(M18-P… and M18-D… only)

ORDER GUIDE

Type		Appearance	Sensing range	Model No. Plastic	Model No. Metal	Output	Terminal
Thru-beam			12m	M18-T120P-PN	M18-T120M-PN	PNP	Wire
				M18-T120P	M18-T120M	NPN	
				M18-T120P-PN-J	M18-T120M-PN-J	PNP	Connector
				M18-T120P-J	M18-T120M-J	NPN	
Retroreflective			2m	M18-R020P-PN	M18-R020M-PN	PNP	Wire
				M18-R020P	M18-R020M	NPN	
				M18-R020P-PN-J	M18-R020M-PN-J	PNP	Connector
				M18-R020P-J	M18-R020M-J	NPN	
	With polarising filters		1.5m	M18-P015P-PN	M18-P015M-PN	PNP	Wire
				M18-P015P	M18-P015M	NPN	
				M18-P015P-PN-J	M18-P015M-PN-J	PNP	Connector
			(with sensitivity adjuster)	M18-P015P-J	M18-P015M-J	NPN	
Diffuse reflective			0.3m	M18-D003P-PN	M18-D003M-PN	PNP	Wire
				M18-D003P	M18-D003M	NPN	
				M18-D003P-PN-J	M18-D003M-PN-J	PNP	Connector
			(with sensitivity adjuster)	M18-D003P-J	M18-D003M-J	NPN	

NOTE: Reflector is not supplied with the retroreflective type sensor. Please select the suitable reflector from the options.

M18

SPECIFICATIONS

Type	Thru-beam		Retroreflective				Diffuse reflective	
					With polarising filters			
	Plastic	Metal	Plastic	Metal	Plastic	Metal	Plastic	Metal
PNP cable	M18-T120P-PN	M18-T120M-PN	M18-R020P-PN	M18-R020M-PN	M18-P015P-PN	M18-P015M-PN	M18-D003P-PN	M18-D003M-PN
NPN cable	M18-T120P	M18-T120M	M18-R020P	M18-R020M	M18-P015P	M18-P015M	M18-D003P	M18-D003M
PNP connector M12	M18-T120P-PN-J	M18-T120M-PN-J	M18-R020P-PN-J	M18-R020M-PN-J	M18-P015P-PN-J	M18-P015M-PN-J	M18-D003P-PN-J	M18-D003M-PN-J
NPN connector M12	M18-T120P-J	M18-T120M-J	M18-R020P-J	M18-R020M-J	M18-P015P-J	M18-P015M-J	M18-D003P-J	M18-D003M-J
Sensing range	12m		0.1m to 2m *1)		0.1m to 1.5m *1)		1cm to 30cm	
Sensing object	ø5mm or more opaque object		ø35mm or more opaque or translucent object		ø7.5mm or more opaque or translucent object		ø5mm or more opaque or translucent object	
Repeatability	0.1mm or less		0.2mm or less		0.2mm or less		0.5mm or less (setting distance: 30cm)	
Supply voltage	10 to 30V DC, reverse polarity protected							
Current consumption	Max. 30mA							
Sensitivity adjuster	Not equipped				Equipped			
Output	Max. 100mA Short circuit protection							
Output operation	Dark-ON / Light-ON (selectable via wiring)							
Response time	2ms		1ms					
Operation indicator	Yellow LED							
Emission indicator	Green LED		—					
Pollution degree	3 (industrial environment)							
Protection	IP67							
Ambient temperature	-25 to +55°C (No dew condensation or icing allowed), storage:-25 to+70°C							
Ambient humidity	35 to 85% RH							
Ambient illuminance	5,000ℓx							
EMC	EN50082-1, EN61000-6-2,:EN50081-2							
Voltage withstandability	500VAC for 1 minute							
Insulation resistance	20MΩ, or more, with 250VDC							
Vibration resistance	10 to 55Hz, 1 cycle/min., double amplitude 0.75mm, 10 min. on 3 axes							
Shock resistance	10G min., 4 times on 3 axes							
Emitting element	Infrared LED				Red LED		Infrared LED	
Material	Plastic type: ABS; metal type: nickel-plated brass							
Optical Material	PMMA plastic							
Connection Method	4-core cable 2m long; or M12 connector (-J)							
Cable extension	Extension up to total 100m possible with 0.34mm², or more, cable (thru-beam type: both emitter and receiver)							
Weight	Max. 210g		Max. 110g					

*1) With M18-RF48

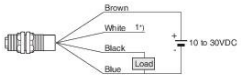
OPTIONS

M18-RF48	Plastic Reflector, 48mm
M18-RF75	Plastic Reflector, 75mm
M18-SPS	M18 Mounting Bracket, Plastic, Flexible
M18-SPF	M18 Mounting Bracket, Plasic, Fixed
M18-ST20	M18 Mounting Bracket, Metal, L Shape 20mm
M18-ST43	M18 Mounting Bracket, Metal, L Shape 43mm
UZZ81220D	2m cable with M12 connector
UZZ81221D	2m cable with M12 connector, ellbow
UZZ81250D	5m cable with M12 connector
UZZ81251D	5m cable with M12 connector, ellbow

NOTE: Reflector is not supplied with the retroreflective type sensor.
Please select the suitable reflector from the options.

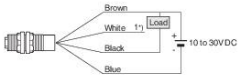
I/O WIRING DIAGRAMS

PNP output type



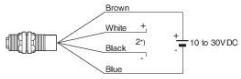
1 10-30VDC
2 Light/Dark
3 0V
4 NO (Normally Open) output

NPN output type



1 10-30VDC
2 Light/Dark
3 0V
4 NO (Normally Open) output

Emitter of thru-beam type

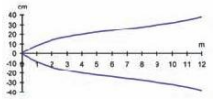


1 10-30VDC
2 Test +
3 0V
4 Test -

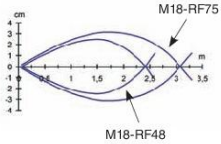
1* With the white wire not connected the diffuse reflective models function in the light mode and the retroreflective and thru-beam models in the dark mode; the light mode can be selected connecting the white wire to +VDC, the dark mode connecting it to 0VDC
2* Emitter off with Test+ on VDC and Test- on 0V

SENSING CHARACTERISTICS (TYPICAL)

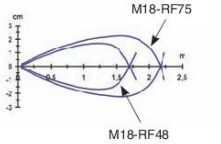
M18-T120□
Thru beam



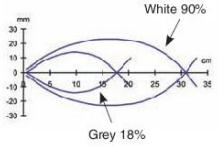
M18-R020□
Retroreflective



M18-P015□
Retroreflective with polarising filter

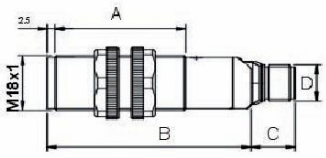


M18-D003□
Diffuse reflective



DIMENSIONS (Unit: mm)

Sensors



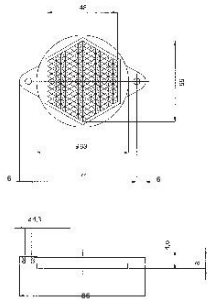
Product	A (mm)	B (mm)	C (mm)	D (mm)
M18-T120□-J (Receiver)	38	57	14.5	M12
M18-T120□-J (Emitter)	43	67	14.5	M12
M18-T120□ (Receiver)	38	57	10	ø4
M18-T120□ (Emitter)	43	67	14.5	ø4
M18-R020□-J	38	57	10	M12
M18-R020□	38	57	10	ø4
M18-P015□-J	43	67	14.5	M12
M18-P015□	43	67	14.5	ø4
M18-D003□-J	43	67	14.5	M12
M18-D003□	43	67	14.5	ø4

NOTE: Reflector is not supplied with the retroreflective type sensor.
Please select the suitable reflector from the options.

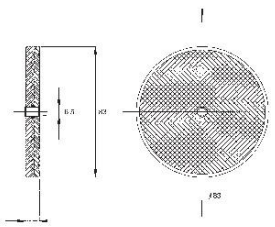
DIMENSIONS (Unit: mm)

Accessories

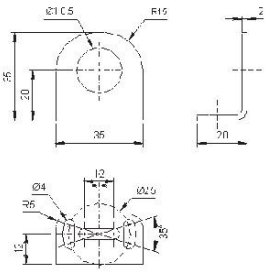
M18-RF48



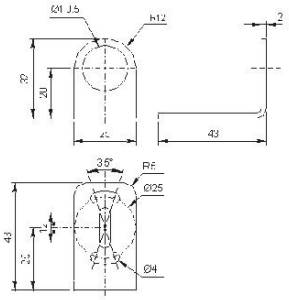
M18-RF75



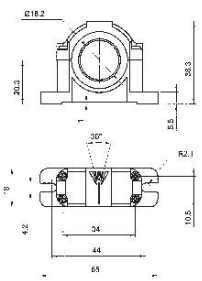
M18-ST20



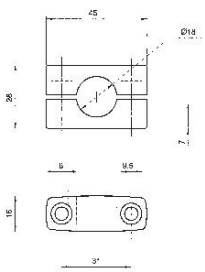
M18-ST43



M18-SPS



M18-SPF



Panasonic Electric Works

Please contact our Global Sales Companies in:

Europe		
▸ Headquarters	**Panasonic Electric Works Europe AG**	Rudolf-Diesel-Ring 2, 83607 Holzkirchen, Tel. (08024) 648-0, Fax (08024) 648-111, www.panasonic-electric-works.com
▸ Austria	**Panasonic Electric Works Austria GmbH** **PEW Electronic Materials Europe GmbH**	Rep. of PEWDE, Josef Madersperger Str. 2, 2362 Biedermannsdorf, Tel. (02236) 26846, Fax (02236) 46133, www.panasonic-electric-works.at Ennshafenstraße 9, 4470 Enns, Tel. (07223) 883, Fax (07223) 88333, www.panasonic-electronic-materials.com
▸ Benelux	**Panasonic Electric Works Sales Western Europe B.V.**	De Rijn 4, (Postbus 211), 5684 PJ Best, (5680 AE Best), Netherlands, Tel. (0499) 372727, Fax (0499) 372185, www.panasonic-electric-works.nl
▸ Czech Republic	**Panasonic Electric Works Czech s.r.o.**	Průmyslová 1, 34815 Planá, Tel. 374 799 990, Fax 374 799 999, www.panasonic-electric-works.cz
▸ France	**Panasonic Electric Works Sales Western Europe B.V.**	French Branch Office, B.P. 44, 91371 Verrières le Buisson CEDEX, Tél. 01 60135757, Fax 01 60135758, www.panasonic-electric-works.fr
▸ Germany	**Panasonic Electric Works Deutschland GmbH**	Rudolf-Diesel-Ring 2, 83607 Holzkirchen, Tel. (08024) 648-0, Fax (08024) 648-555, www.panasonic-electric-works.de
▸ Ireland	**Panasonic Electric Works UK Ltd.**	Dublin, Tel. (01) 4600969, Fax (01) 4601131, www.panasonic-electric-works.co.uk
▸ Italy	**Panasonic Electric Works Italia s.r.l.**	Via del Commercio 3-5 (Z.I. Ferlina), 37012 Bussolengo (VR), Tel. (045) 6752711, Fax (045) 6700444, www.panasonic-electric-works.it
	PEW Building Materials Europe s.r.l.	Piazza della Repubblica 24, 20154 Milano (MI), Tel. (02) 29005391, Fax (02) 29003466, www.panasonic-building-materials.com
▸ Nordic Countries	**Panasonic Electric Works Nordic AB** **PEW Fire&Security Technology Europe AB**	Sjöängsvägen 10, 19272 Sollentuna, Sweden, Tel. (08) 59476680, Fax (08) 59476690, www.panasonic-electric-works.se Citadellsvägen 23, 21118 Malmö, Tel. (040) 6977000, Fax (040) 6977099, www.panasonic-fire-security.com
▸ Poland	**Panasonic Electric Works Europe AG**	Przedstawicielstwo w Polsce, Al. Krakowska 4/6, 02-284 Warszawa, Tel. 22 338-11-33, Fax 22 338-12-00, www.panasonic-electric-works.pl
▸ Portugal	**Panasonic Electric Works España S.A.**	Portuguese Branch Office, Avda Adelino Amaro da Costa 728 R/C J, 2750-277 Cascais, Tel. (21) 4812520, Fax (21) 4812529
▸ Spain	**Panasonic Electric Works España S.A.**	Barajas Park, San Severo 20, 28042 Madrid, Tel. (91) 3293875, Fax (91) 3292976, www.panasonic-electric-works.es
▸ Switzerland	**Panasonic Electric Works Schweiz AG**	Grundstrasse 8, 6343 Rotkreuz, Tel. (041) 7997050, Fax (041) 7997055, www.panasonic-electric-works.ch
▸ United Kingdom	**Panasonic Electric Works UK Ltd.**	Sunrise Parkway, Linford Wood, Milton Keynes, MK14 6LF, Tel. (01908) 231555, Fax (01908) 231599, www.panasonic-electric-works.co.uk
North & South America		
▸ USA	**PEW Corporation of America**	629 Central Avenue, New Providence, N.J. 07974, Tel. 1-908-464-3550, Fax 1-908-464-8513, www.pewa.panasonic.com
Asia Pacific / China / Japan		
▸ China	**Panasonic Electric Works (China) Co., Ltd.**	Level 2, Tower W3, The Towers Oriental Plaza, No. 2, East Chang An Ave., Dong Cheng District, Beijing 100738, Tel. (010) 8518-5988, Fax (010) 8518-1297
▸ Hong Kong	**Panasonic Electric Works (Hong Kong) Co., Ltd.**	RM1205-9, 12/F, Tower 2, The Gateway, 25 Canton Road, Tsimshatsui, Kowloon, Hong Kong, Tel. (0852) 2956-3118, Fax (0852) 2956-0398
▸ Japan	**Matsushita Electric Works, Ltd.**	1048 Kadoma, Kadoma-shi, Osaka 571-8686, Japan, Tel. (06) 6908-1050, Fax (06) 6908-5781, www.mew.co.jp/e-acg/
▸ Singapore	**Panasonic Electric Works Asia Pacific Pte. Ltd.**	101 Thomson Road, #25-03/05, United Square, Singapore 307591, Tel. (06255) 5473, Fax (06253) 5689

Panasonic®

REFERENCIAS

[1] Enrique Mandado, Jorge Marcos Acevedo, Celso Fdz., Jose I. Armesto, Autómatas programables y sistemas de automatización, 1ª ed., Marcombo, 2009.

[2] Karl Heinz John, Michael Tiegelkamp, IEC 61131-3: Programming Industrial Automation Systems, 2ª ed., Springer, 2010.

[3] Joan Domingo Peña, Juan Gámiz Caro, Antoni Grau i Sakdes, Herminio Martínez García, Introducción a los autómatas programables, 1ª ed., UOC, 2003.

[4] A. Simon, AUTÓMATAS PROGRAMABLES. Programación, automatismos y lógica programada, 3ª ed., Paraninfo, 1995.

[5] Josep Ballcels, José Luis Romeral, Autómatas programables, 1ª ed., Marcombo, 2000.

[6] G. Michel, Autómatas programables industriales. Arquitectura y aplicaciones, 1ª ed., Marcombo, 1990.

[7] «Beckhoff,» [En línea]. Available: http://infosys.beckhoff.com/espanol.php?content=../content/1034/tcplccontrol/html/tcplcctrl_languages%20st.htm&id . [Último acceso: Agosto 2016].

[8] «Softwaredoit - Depuracion,» [En línea]. Available: https://www.softwaredoit.es/definicion/definicion-depuracion.html . [Último acceso: Agosto 2016].

[9] «InfoPLC - Tutorial Práctico del autómata M340 y Unity Pro 3.0,» [En línea]. Available: http://www.infoplc.net/descargas/170-schneider-electric/automatas/m340/355-tutorial-practico-del-automata-m340-y-unity-pro-30. [Último acceso: Agosto 2016].

[10] «InfoPLC - Guía Rápida del software Unity Pro,» [En línea]. Available: http://www.infoplc.net/descargas/182-schneider-electric/software-instrucciones/unity-pro/330-guia-rapida-del-software-unity-pro. [Último acceso: Agosto 2016].

[11] Schneider Electric, «Unity Pro - Modalidades de funcionamiento,» 2015.

[12] Schneider Electric, «Unity Pro - Lenguajes y estructura del programa. Manual de referencia,» 2015.

[13] Schneider Electric, «Modicom M340 con Unity Pro - Procesadores, bastidores y módulos de fuentes de alimentación - Manual de configuración,» 2015.

[14] Schneider Electric, «Unity Pro - Palabras y bits de sistema. Manual de referencia,» 2015.

[15] Schneider Electric, «Unity Pro - Lenguajes y estructura del programa. Manual de referencia,» 2009.

[16] Schneider Electric, «Unity Pro - TCP Open. Block Library,» 2015.

[17] «Universidad de Oviedo - Ingenieria de sistemas y automática. Presentación IEC 61131-3,» [En línea]. Available: http://isa.uniovi.es/docencia/IngdeAutom/transparencias/Pres%20IEC%2061131.pdf. [Último acceso: Agosto 2016].

[18] «PLCOpen - Intro IEC 61131-3,» [En línea]. Available: http://www.plcopen.org/pages/pc2_training/introductions_in_spanish_and_portugese/downloads/intro_iec_61131_3_spanish.doc. [Último acceso: Agosto 2016].

[19] «InfoPLC - Introducción al estándar IEC 61131-3,» [En línea]. Available: http://www.infoplc.net/files/documentacion/estandar_programacion/infoPLC_net_Intro_estandar_IEC_61131-3.pdf. [Último acceso: Agosto 2016].

[20] «ProfesorMolina - Tecnología,» [En línea]. Available: http://www.profesormolina.com.ar/tecnologia/plc/index.htm. [Último acceso: Agosto 2016].

[21] «Panasonic - FPWin,» [En línea]. Available: https://panasonic-electric-works.com/es/control-fpwin-pro.htm#. [Último acceso: Agosto 2016].

[22] Schneider Electric, «Guía de soluciones de automatización y control industrial».

[23] Eshed Robotec, «ASRS Manual de usuario,» 1996.

[24] Karl-Heinz John, Michael Tiegelkamp, IEC 61131-3: Programming Industrial Autoation Systems, 2ª ed., 2010.

Lista de abreviaturas

AFCET: Asociación Francesa para la Cibernética Económica y Técnica

CAN: Controller Area Network

CanOpen: Protocolo comunicación para uso industrial basado en el bus CAN

DC: Duty Cycle

DDT: Derived Data Type

DFB: Derived Function Block

DIR: Direction

EF: Elementary Function

EFB: Elementary Function Block

EN: Enable

ENO: Enable Output

EP: Elaboración Propia

FB: Function Block

FBD: Function Block Diagram

FFB: Término genérico para los tipos de bloques funcionales en Unity Pro

GND: Ground

GRAFCET: Grafo de Control Etapa Transición

HTML: HyperText Markup Language

IEC: International Electrotechnical Comission

IL: Instruction List

LD: Ladder

MODBUS: Bus de comunicación Modicom

MODICOM: Modular Digital Controller

PaP: Paso a Paso

PLC: Programable Logic Controller

POU: Program Organization Unit

PUL: Pulse

SFC: Sequential Function Chart

ST: Structured Text

TP: Time Pulse